AFIT/EN/ENG/99M-09

Multiple Model Adaptive Estimation Using Filter Spawning

THESIS

Kenneth A. Fisher, B.S.E.E.
2Lt, USAF

AFIT/EN/ENG/99M-09

AFIT/EN/ENG/99M-09

Multiple Model Adaptive Estimation Using Filter Spawning

THESIS

Presented to the Faculty of the Graduate School of Engineering of the Air Force Institute of Technology Air University In Partial Fulfillment for the Degree of

Masters of Science

Specialization in: Electrical Engineering

Kenneth A. Fisher, B.S.E.E.

2Lt, USAF

Air Force Institute of Technology

Wright-Patterson AFB, Ohio

March, 1998

AFIT/EN/ENG/99M-09

Multiple Model Adaptive Estimation Using Filter Spawning

Kenneth A. Fisher, B.S.E.E.

2Lt, USAF

Approved:

Peter S. Maybeck
Dr. Peter S. Maybeck
Committee Chairman

March 8, 1999
Date

M. Pachter
Dr. Meir Pachter
Committee Member

March 8, 1999
Date

Disclaimer

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

Acknowledgments

I would like to thank those who have helped me complete this research. Dr. Peter Maybeck, my thesis advisor, was supportive and excited throughout the process; he made me feel my research and myself were important. Dr. Meir Pachter, my committee member, provided a sound background in flight control and system identification to make sure Dr. Maybeck and I did not overlook anything. I would also like to recognize the other professors in the department that took interest in my research: Dr. Mikel Miller (Maj, USAF) and Dr. John Raquet (Capt, USAF).

I would like to thank AFRL/VACC for their sponsorship in this research. I would like to thank Capt. Odell Reynolds (PhD) and Bob Smith for their UNIX and FORTRAN knowledge and resources. I would like to thank Don Smith for his "go-kart" stories, bowling, and technical support. And "gandc", my friends, trivia champions, campers, and Canadians, thank you for the intellectual stimulation and Homer sound-byte relief.

I would also like to thank the Ohio Northern University professors for the preparation I received. I would like to thank Promise Keepers at AFIT, for their prayers and weekly bible study. Finally, I would like to thank my wife, Amy, for taking care of me, loving me, praying for me, encouraging me, and just being there to smile.

And just as I have said when taking vacations with my father to desolate places, "It was an enjoyable time, and I am glad it's over."

Table Of Contents

List of Figures

List of Tables

Abstract

Multiple Model Adaptive Estimation with Filter Spawning is used to detect and estimate partial actuator failures on the VISTA F-16. The truth model is a full six-degree-of-freedom simulation provided by Calspan and General Dynamics, including the aircraft's nonlinear equations of motion, fourth order actuator models, the complete Block 40 flight control system, and the aileron-to-rudder interconnect. The design models are chosen as 13-state linearized models, including first order actuator models. Actuator failures are incorporated into the truth model and design model assuming a "failure to free stream". Filter Spawning is used to include additional filters with partial actuator failure hypotheses into the Multiple Model Adaptive Estimation (MMAE) bank. The spawned filters are based on varying degrees of partial failures (in terms of effectiveness) associated with the complete-actuator-failure hypothesis with the highest conditional probability of correctness at the current time. Thus, a blended estimate of the failure effectiveness is found using the filters' estimates based upon a no-failure hypothesis (or, an effectiveness of 100%), a complete actuator failure hypothesis (or, an effectiveness of 0%), and the spawned filters' partial-failure hypotheses. This yields substantial precision in effectiveness estimation, compared to what is possible without spawning additional filters, making partial failure adaptation a viable methodology in a manner heretofore unachieved. The failure effectiveness estimate is *refined* based on the empirical relationship between the effectiveness estimate and the true effectiveness. The refined estimate is found to yield accurate results two seconds after failure detection in most effectiveness regions. The refined estimate is suitable for use in applying control via modified Control Redistribution to handle complete and partial actuator failures, and in considering dual failures.

Multiple Model Adaptive Estimation Using Filter Spawning

Chapter 1 - Introduction

1.1 Chapter Overview

This thesis presents Multiple Model Adaptive Estimation with Control Redistribution and Filter Spawning applied to the detection and estimation of partial actuator failures on the Variable-Stability In-flight Simulator Test Aircraft (VISTA) F-16. Section 1.2 motivates the work of this thesis. Then Section 1.3 describes the problem definition, and Section 1.4 summarizes the assumptions made. Finally, Section 1.5 outlines the thesis format.

1.2 Motivation

The United States Air Force recognizes the need for fault-tolerant and survivable flight control systems in its aircraft. In particular, flight control systems should be able to detect and estimate actuator and sensor failures, and reconfigure the control law of the aircraft to obtain the least degradation in performance as possible. The success of an algorithm that can detect and estimate failures may eliminate the need for layers of redundant sensors and actuators, reducing aircraft weight and cost. The success of an algorithm that can control an aircraft in the face of failures may improve performance during sensor failures (by using a "better" reconstructed measurement vector estimate, rather than using the raw measurements themselves, in forming the control) and reduce the hazard associated with actuator failures (either the loss of the aircraft or the loss of mission-essential aircraft performance).

Multiple Model Adaptive Estimation with Control Redistribution (MMAE/CR) has been chosen based on its detection, estimation, and control performance for *complete* actuator and sensor failures [18, 41, 42]. In considering *partial* actuator failures, filter spawning has been chosen to include additional filters in the MMAE bank based on partial actuator failure hypotheses at the rec-

ommendation of Clark [8], based on the inadequate performance found using an MMAE bank based only on complete-failure and no-failure hypotheses (i.e., without spawning).

1.3 Problem Statement

A long line of research [8, 10, 11, 18, 31, 32, 41, 42] has explored the detection and estimation of actuator and sensor failures on the VISTA F-16. Adequate performance has been achieved for complete sensor failures, complete actuator failures, and any dual combination thereof [18, 41, 42]. Dual failures have been considered for the case in which one of the failures is a partial actuator failure; however, the performance demonstrated did not warrant its implementation. This research seeks to detect and estimate partial actuator failures through an alternate conceptualization, called Filter Spawning. Dual failures and control aspects will not be explored explicitly in this research; rather, these aspects are left for future consideration. In summary, the detection and estimation of partial actuator failures will be considered using MMAE with Filter Spawning.

1.4 Assumptions

The "real world" is the VISTA F-16. The "truth model" is a full six-degree-of-freedom simulation provided by Calspan and General Dynamics. It is assumed that the "truth model" is a very accurate representation of the "real world".

Actuator failures are modelled as a "failure to free stream" in the "truth model" as well as in the "design model" used for the basis of the filter algorithm. It is assumed that "failure to free stream" actuator failures occur in the real world, and that actuator failures that change the plant matrix (such as battle damage) or other actuator failures can be approximated with such a model.

1.5 Thesis Format

This thesis has a six-chapter format. Chapter 1, this chapter, provides an introduction to the thesis topic and format. Chapter 2 presents the history of MMAE, the theory of MMAE, and modifications to MMAE that enhance performance and on-line implementation. Chapter 2 presents these topics in a general environment; that is, most of the history, theory, and modifications for

implementation are *not* dependant on this specific application. Chapter 3 first presents the "real world", the "truth model", the "design model", and the "failure models" incorporated into this research. With the models at hand, Control Redistribution is explained as it is applied to this research at the end of Chapter 3. Chapter 4 presents MMAE with Filter Spawning in general and as it is applied to this research. Chapter 5 presents the performance results of MMAE with Filter Spawning applied to the detection and estimation of partial actuator failures. Chapter 6 reviews the problem, the methodology, and the results. Appendix A is a supplement to the description of the design model discussed in Chapter 3. Appendices B and C are supplements to the results presented in Chapter 5.

1.6 Chapter Summary

This chapter has provided an introduction to the detection and estimation of partial actuator failures using MMAE with Filter Spawning. The motivation, problem definition, assumptions, and thesis format have been discussed to establish the context of the research that has been conducted.

Chapter 2 - Concept and Algorithm Development

2.1 Chapter Overview

This chapter provides a basic overview of Multiple Model Adaptive Estimation (MMAE). It starts with the historical background of MMAE, including early concept development and contributions from the Air Force Institute of Technology. The remaining two sections present the theory of MMAE and modifications to MMAE for practical implementation. Presented this way, it should be clear what ideas have theoretical merit and what ideas have been found through research to enhance implementation.

2.2 Multiple Model Adaptive Estimation History

2.2.1 Early Contributions

2.2.1.1 Theory. In 1965, Magill [21] first presented the idea now known as Multiple Model Adaptive Estimation (MMAE). To form the basis of MMAE, Magill proposed to construct elemental filters, each based on a different hypothesis about the real world system's parameter values or other attributes. Then, using the conditional probability of each hypothesis being correct (conditioned on the measurements actually observed) as a weighting, the algorithm forms a blended estimate. In 1976, Lainiotis [17] first presented the idea now known as Multiple Model Adaptive Control (MMAC). Lainiotis cascaded elemental controllers with the elemental filters to produce analogously an "optimal adaptive controller". At the time, MMAE and MMAC algorithms were mostly theoretical exercises for two reasons. First, implementation was computationally infeasible without parallel processors. Second, the recursive nature of the probability calculations would eventually cause some of the computed probabilities to go to zero; thus, the system would become insensitive to change.

Chang and Athans [7] contributed optimality conditions for MMAE. They proved that the Bayesian (blended) method was only optimal provided the true parameter value exactly matched one of the hypothesized parameter values. Further, it was shown that if the true parameter value did not match the modelled parameter value, the MMAE was suboptimal. Tugnait [46] showed that,

for Markov-1 parameter processes[1], the algorithm is again suboptimal. Although the necessary conditions for optimality are not met in most cases, including this research effort, this is not overly restrictive. Every designer must admit that a finite system model fails to represent the real world fully. Thus, whether an algorithm can claim optimality or not with respect to a (perhaps very artificial) set of mathematical modeling assumptions, it must be tested against a *real* world truth model to determine true performance.

2.2.1.2 Implementation. In 1977, Athans *et al* [1] developed a *practical implementation* for the MMAE algorithm. This application presented a flight control problem in which the flight condition was estimated. Athans *et al* introduced methods to enhance implementation and also encountered problems with MMAE.

Two methods were used to enhance implementation of MMAE. First, a lower bound on the probability calculations was introduced, discussed in Section 2.4.1. This decreased response times (i.e., the estimation algorithm was more agile) and avoided the problem of probabilities going to zero. This was a very useful alternative to a full-scale Markov model for parameter dynamics, which could accomplish the same benefits, but with significantly greater on-line computational expense and considerably more complicated effort required of the designer. Second, a weighted average (or Bayesian method) was used to calculate the control signal for the MMAC rather than using the one control signal corresponding to the filter with the highest probability (using the Maximum A Posteriori, or MAP, method). These two approaches will be discussed in Section 2.3.2.3.

Two problems were found with MMAE. First, "beta dominance" impaired flight condition estimation, as discussed in Section 2.4.2. Second, a dither signal, or test input, was needed to excite the system and thereby improve system identification, as discussed in Section 2.4.4.

Although this application formed the basis for implementing MMAE and MMAC, it suffered two limitations. First, the aircraft chosen[2] was inherently stable; therefore, the success of adaptive

[1]Markov-1 processes are time-varying processes for which knowledge of the system states at the current time step is sufficient to completely characterize the system states at the next time step.

[2]The aircraft was an F-8 "Crusader" fighter jet.

control was not particularly noteworthy[3]. Second, available digital flight computer limitations could not allow a full implementation. The algorithm only used four of the 16 designed filters at a relatively slow 8 Hz sample rate (as compared to the 64 Hz sample rate of current research).

2.2.2 Contributions at the Air Force Institute of Technology

The Air Force Institute of Technology (AFIT) is a major contributor in the MMAE field. AFIT has *applied* MMAE algorithms to various problems, including detection of failed sensors and/or actuators [8,10,11,18,31,32,41,42], adaptive target tracking [30], adaptive control of flexible space vehicles [14,26], detection and compensation of interference/jamming and spoofing of GPS-aided inertial navigation systems [48], adaptive head motion predictors in virtual environment flight simulators [29], and adaptive control of robot arms [45].

AFIT has also researched areas in beta dominance, moving-bank filters, and discretization methods. Recall that Athans *et al* encountered beta dominance problems in their flight control algorithm implementation (See Section 2.2.1.2). In 1985, Maybeck and Suizu [29] addressed the adverse effect of the beta term (See Equations (2.21) and (2.22) for a full discussion of the "beta" term). For their application, removing the β term, i.e., equating it to unity, remarkably improved results. This was also shown to be particularly beneficial in the application of MMAE and MMAC to the detection of sensor and actuator failures in flight control systems by Stevens [28,43]. Stevens in 1989 and Menke [31,32] in 1993 investigated "scalar residual monitoring" as an alternate solution to the beta dominance problem as well. Beta dominance will be discussed in full as it applies to this research in Section 2.4.2. In 1987, Maybeck and Hentz [25] considered reducing the total number of on-line filters in a multiple model algorithm by means of a moving bank of filters. The moving-bank algorithm *logic* is written in such a way that the parameter values of the filters' hypotheses seem to *move* throughout the parameter space. AFIT has furthered moving-bank application and theory [47]. A particular form of moving-bank algorithm, known as the hierarchical structure algorithm [8,10,11,18,28,31,32,43] is especially pertinent to failure detection in flight control systems;

[3]To be fair, that was not the fault of the authors – their research was applied to an IEEE-declared benchmark problem to which design teams were requested to apply adaptive controllers.

it will be described more fully in Section 2.3.4. AFIT has also investigated proper discretization of parameter spaces [25,39,40] as well as the proper discretization of moving-bank algorithms [47].

2.2.3 AFIT's History in Applying MMAE to Flight Control

The U.S. Air Force has always recognized the need for fault detection and estimation in its flight control systems. Thus, AFIT has pursued an MMAE application to failure detection and estimation of sensors and actuators on current fighter aircraft. Through the years, the implementation has been applied to the STOL F-15, unmanned aerospace vehicles, and the VISTA F-16. Research has developed from initial modelling to single full failures to dual full failures, to single and dual partial failures. Presented here are the most recent contributors, namely Menke, Eide, Stepaniak, Lewis, and Clark.

In 1992, Menke [31,32] performed analysis of the Variable Stability, In-flight Simulator Test Aircraft (VISTA) F-16 in a low dynamic pressure flight condition (0.4 Mach, 20,000 feet) using a Dryden wind model developed by Pogoda [24,37]. The low dynamic pressure condition was chosen because it is the *most* challenging for failure detection. Menke considered various types of sensor failures, including loss of signal, increased noise levels, and biases. He established the need for a sinusoidal dither signal in the absence of pilot command, used the hierarchical structure for dual failure detection, and used scalar residual monitoring to reduce beta dominance effects and to enhance performance above that with MMAE alone by detecting dither sinusoids in the residuals of elemental filters based on incorrect failure hypotheses.

In 1994, Eide [10,11] implemented the six degree-of-freedom Simulation Rapid-prototyping Facility (SRF) simulation for the VISTA F-16. The SRF VISTA simulation includes complete nonlinear lateral/directional and longitudinal dynamics, rather than the linearized models of earlier efforts. The F-16's entire Block 40 flight control system, including the aileron-rudder interconnect, is encoded as the controller, in order to provide a very realistic simulation. All actuator saturations and rate limits are also included in the simulation. This SRF implementation became the standard truth model throughout the ensuing research.

In 1995, Stepaniak [41,42] contributed an alternative to MMAC and a refinement of MMAE-based control, called Control Redistribution (CR), and refined an appropriate dither signal. He successfully demonstrated MMAE's ability to detect and compensate for single, complete failures of any actuator or sensor.

Up to this point, ability to detect multiple failures suffered due to impaired dither signals if the first failure was an actuator. In 1996, Lewis [18] contributed an appropriate *redistributed* dither once an actuator failed. He successfully demonstrated MMAE's ability to detect dual, complete failures of all actuator and sensor combinations.

In 1997, Clark [8] contributed a blending approach to detecting partial failures. Clark blended complete failure and no failure results to obtain an estimate for use in the case of a partial failure. In most cases, the first partial failure was detected with some adequacy, but performance was not sufficient to warrant actual implementation. The current research seeks enhanced detection through an alternate conceptualization, in order to yield a practical system worthy of flight test.

2.3 MMAE in Theory

Kalman filters estimate states of a dynamic system, assuming a particular parameter vector value for the system (See Section 2.3.1). In the current application, this parameter vector value corresponds to the failure status of all sensors and actuators, but other applications can have very different possible parameters. One could build many of these filters, each assuming a different parameter vector value for the system model within a parameter space. Then, each Kalman filter would provide a different estimate of the system states based on the different system models. Using the information provided by such a scheme, Multiple Model Adaptive Estimation (MMAE) determines a conditional probability associated with each filter (and its corresponding estimate), conditioned on the measurements actually observed from the real-world system. MMAE can then provide a state estimate over a range of parameter vector values within the parameter space. The conditional probabilities can be used to obtain a parameter vector estimate as well. Section 2.3.2 will discuss this in more detail.

Control authority could be determined by augmenting controllers to each Kalman filter within the MMAE structure (yielding a Multiple Model Adaptive Controller, or MMAC) or by augmenting one controller outside the MMAE loop (yielding what is called MMAE-based control). Both are discussed in Section 2.3.3.

To take MMAE one step further, one could assemble multiple MMAE algorithms, or banks, based on various families of parameter vector values. Using only one of these banks at a time, logic could incorporate information obtained from the current bank to determine which bank (and which family of parameter vector values) the algorithm should use next. This is the basis for the hierarchical structure, discussed in Section 2.3.4.

2.3.1 Kalman Filters

The Kalman filter is an optimal estimator of state variables, based upon a particular parameter vector value to describe the system model. The advantage of this estimator is that the estimate uses information about the model of the system as well as measurements of the system. The Kalman filter completes this in two stages. First, the filter computes a *prediction* of the system states based on the *imperfect* dynamics model it was given. This is called the propagate stage. Second, the filter refines the system state estimates through the use of *noise corrupted* measurements. This is called the update stage. A complete derivation can be found in *Stochastic Models, Estimation, and Control Volume 1* [22]; only the resulting equations will be summarized here [41,42].

The dynamics model is assumed to be a linear time-invariant, discrete-time system of the form

$$\mathbf{x}(t_{i+1}) = \Phi\mathbf{x}(t_i) + \mathbf{B}_d\mathbf{u}(t_i) + \mathbf{G}_d\mathbf{w}_d(t_i) \tag{2.1}$$

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.2}$$

where $\mathbf{x}$ is the vector of system states, $\mathbf{u}$ the control input vector, $\mathbf{z}$ the measurement vector, and where dynamics driving noise, $\mathbf{w}_d(t_i)$, and measurement corruption noise, $\mathbf{v}(t_i)$, are discrete-time, white Gaussian noises with the statistics:

$$E\left[\mathbf{w}_d(t_i)\right] = \mathbf{0} \qquad E\left[\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_i)\right] = \mathbf{Q}_d \qquad E\left[\mathbf{w}_d(t_i)\mathbf{v}^T(t_i)\right] = \mathbf{0}$$
$$E\left[\mathbf{v}(t_i)\right] = \mathbf{0} \qquad E\left[\mathbf{v}(t_i)\mathbf{v}^T(t_i)\right] = \mathbf{R} \tag{2.3}$$

The filter is propagated forward by

$$\hat{\mathbf{x}}(t_{i+1}^-) = \Phi\hat{\mathbf{x}}(t_i^+) + \mathbf{B}_d\mathbf{u}(t_i) \tag{2.4}$$

$$\mathbf{P}(t_{i+1}^-) = \Phi\mathbf{P}(t_i^+)\Phi^T + \mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^T \tag{2.5}$$

starting from the initial conditions, $\hat{\mathbf{x}}(t_0)$ and $\mathbf{P}(t_0)$. The filter is updated by

$$\mathbf{A}(t_i) = \mathbf{H}\mathbf{P}(t_i^-)\mathbf{H}^T + \mathbf{R} \tag{2.6}$$

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T\mathbf{A}^{-1}(t_i) \tag{2.7}$$

$$\mathbf{r}(t_i) = \mathbf{z}_i - \mathbf{H}\hat{\mathbf{x}}(t_i^-) \tag{2.8}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\mathbf{r}(t_i) \tag{2.9}$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}\mathbf{P}(t_i^-) \tag{2.10}$$

where the $-$ and $+$ superscripts indicate before and after the measurement is taken, respectively[4]. The residual vector, $\mathbf{r}(t_i)$, can be shown to be zero-mean, with the filter-predicted covariance of the residual vector described by $\mathbf{A}_k(t_i)$, as computed in Equation (2.6), if the filter is based upon the correct hypothesized model.

It is important to emphasize what information a Kalman filter provides. It provides an estimate of the system states, $\hat{\mathbf{x}}(t_i^+)$, the filter-computed error covariance of the system state estimate, $\mathbf{P}(t_i^+)$, the residual vector, $\mathbf{r}(t_i)$, and the filter-computed covariance of the residual vector, $\mathbf{A}(t_i)$. This information is useful when forming a Multiple Model Adaptive Estimation algorithm, discussed next.

2.3.2 Multiple Model Adaptive Estimation

Recall that a Kalman filter is based on one particular model of the system. Further, the residual vector, $\mathbf{r}(t_i) = \mathbf{z}_i - \mathbf{H}\hat{\mathbf{x}}(t_i^-)$, is a measure of how well the modelled system matches the real world measurements. In MMAE, a bank, or set, of Kalman filters is built, each with a different hypothesis of the system model. If all the filters are given the same measurement vector but are based on different system models, *residual monitoring* can be used to determine which filter's system model best matches the true system model.

[4]Some authors represent t_i^- as $t_i|t_{i-1}$ and t_i^+ as $t_i|t_i$.

2.3.2.1 A Bank of Kalman Filters. To do this, suppose the bank contains K filters, where each of the K filters' system model is based on a different realization of the parameter vector $\mathbf{a}$. For instance, the k^{th} filter would be based on the system model with the characteristics of $\mathbf{a}_k$. Then, each of the K filters is propagated and updated by

$$\hat{\mathbf{x}}_k(t_{i+1}^-) = \Phi_k \hat{\mathbf{x}}_k(t_i^+) + \mathbf{B}_{dk}\mathbf{u}(t_i) \tag{2.11}$$

$$\mathbf{P}_k(t_{i+1}^-) = \Phi_k \mathbf{P}_k(t_i^+)\Phi_k^T + \mathbf{G}_{dk}\mathbf{Q}_{dk}\mathbf{G}_{dk}^T \tag{2.12}$$

$$\mathbf{A}_k(t_i) = \mathbf{H}_k \mathbf{P}_k(t_i^-)\mathbf{H}_k^T + \mathbf{R}_k \tag{2.13}$$

$$\mathbf{K}_k(t_i) = \mathbf{P}_k(t_i^-)\mathbf{H}_k^T \mathbf{A}_k^{-1}(t_i) \tag{2.14}$$

$$\mathbf{r}_k(t_i) = \mathbf{z}_i - \mathbf{H}_k \hat{\mathbf{x}}_k(t_i^-) \tag{2.15}$$

$$\hat{\mathbf{x}}_k(t_i^+) = \hat{\mathbf{x}}_k(t_i^-) + \mathbf{K}_k(t_i)\mathbf{r}_k(t_i) \tag{2.16}$$

$$\mathbf{P}_k(t_i^+) = \mathbf{P}_k(t_i^-) - \mathbf{K}_k(t_i)\mathbf{H}_k\mathbf{P}_k(t_i^-) \tag{2.17}$$

starting from the initial conditions, $\hat{\mathbf{x}}(t_0)$ and $\mathbf{P}(t_0)$, where the following notation is introduced:

$$\Phi_k = \Phi(\mathbf{a}_k)$$

$$\mathbf{B}_{dk} = \mathbf{B}_d(\mathbf{a}_k)$$

$$\mathbf{G}_{dk} = \mathbf{G}_d(\mathbf{a}_k)$$

$$\mathbf{Q}_{dk} = \mathbf{Q}_d(\mathbf{a}_k)$$

$$\mathbf{H}_k = \mathbf{H}(\mathbf{a}_k)$$

$$\mathbf{R}_k = \mathbf{R}(\mathbf{a}_k)$$

The residuals are evaluated by the conditional hypothesis probability values. The conditional hypothesis probability of the k^{th} filter at time t_i is the probability at time t_i that the random variable representing the possible system characteristics, $\mathbf{a}$, is the realization of the variable $\mathbf{a}$ hypothesized by the k^{th} elemental filter, $\mathbf{a}_k$, given the time history of the measurements up to and including that taken at time t_i, $\mathcal{Z}(t_i) = \left[\mathbf{z}(t_1)^T \vdots \mathbf{z}(t_2)^T \vdots \cdots \vdots \mathbf{z}(t_i)^T\right]^T$. This is expressed as

$$p_k(t_i) = Prob(\mathbf{a} = \mathbf{a}_k | \mathcal{Z}(t_i) = \mathcal{Z}_i) \tag{2.18}$$

It has been shown [21,23] that probabilities can be computed as

$$p_k(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a},Z(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1})p_k(t_{i-1})}{\sum_{j=1}^{K} f_{\mathbf{z}(t_i)|\mathbf{a},Z(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathcal{Z}_{i-1})p_j(t_{i-1})} \tag{2.19}$$

Note the denominator acts as a scaling factor to ensure that the probability is properly defined in the sense that

$$p_k(t_i) \geq 0 \quad \text{for all} \quad k \quad \text{and} \quad \sum_{k=1}^{K} p_k(t_i) = 1 \tag{2.20}$$

The conditional density function in Equation (2.19) is recognized as the density function for the current measurement based on the assumed parameter value and the observed time history of all previous measurements, and can be expressed as:

$$f_{\mathbf{z}(t_i)|\mathbf{a},Z(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1}) = \beta_k(t_i)e^{[-\frac{1}{2}\mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i)]} \tag{2.21}$$

where

$$\beta_k(t_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\mathbf{A}_k(t_i)|^{\frac{1}{2}}} \tag{2.22}$$

Careful inspection reveals that Equation (2.21) is actually the density function for the k^{th} Kalman filter's residuals, which are white and Gaussian with zero mean and covariance $\mathbf{A}_k$ if the assumed parameter value, $\mathbf{a}_k$, is correct. For convenience, define the quadratic term in Equation (2.21) to be the likelihood quotient,

$$L_k(t_i) = \mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i) \tag{2.23}$$

To illustrate this, assume that the k^{th} filter's hypothesis best matches the true parameter value. Its residuals will have a mean squared value most in consonance with the corresponding filter-predicted covariance, and therefore the likelihood quotient will approach m (the dimension of the measurement vector, $\mathbf{z}(t_i)$, and the residual vector, $\mathbf{r}_k(t_i)$). Compare this to the likelihood quotient for the mismatched filters. As the measurements deviate farther from the filter-predicted values, both the residual vector and the likelihood quotient increase in magnitude. Referring back to Equation (2.21), when multiplied by the negative scalar coefficient, the argument of the exponential term becomes increasingly negative, and so the density calculations for the mismatched filters quickly drop off toward zero. On the other hand, the exponential terms corresponding to the best matched

filters approach a finite positive value, $e^{-m/2}$. The terms corresponding to the more closely matched filters will then dominate when the probability is calculated using Equation (2.19).

2.3.2.2 MMAE Convergence. The ability of the MMAE to converge to the correct hypothesis is guaranteed theoretically [9, 15, 34] under certain specified conditions. These proofs, however, require that the modelled characteristic $\mathbf{a}_k$ match the true characteristic perfectly. The only assurance of convergence if this is not the case (i.e., always) is that the algorithm will converge to the modelled characteristic closest to the true characteristic in the Baram distance sense [2–4]. These proofs give no guarantee to the convergence *rate*. For this thesis effort, empirical studies of failures in flight control systems have shown that the filters converge in a reasonable amount of time (on the order of seconds or fractions of seconds).

2.3.2.3 Forming an MMAE Estimate. In MMAE, there are two standard ways to obtain a state estimate. Each method uses the elemental probabilities calculated in Equation (2.19). The difference between the two methods lies in how the *best* parameter and state vector estimates are chosen. The Bayesian method blends the hypothesized parameter vectors and state estimates of each filter based on their associated filter's conditional probability to form a conditional *mean* estimate, while the maximum a posteriori (MAP) method takes a maximum likelihood approach to form a conditional *mode* estimate.

First, the Bayesian method uses the conditional *mean* as its *best* estimates of the system parameter vector and system state vector. The Bayesian method determines the best system parameter vector estimate, $\hat{\mathbf{a}}_{MMAE}^{Bayesian}(t_i)$, using a weighted *blending* of the elemental filters' hypothesized system parameter vectors. Analogously, the Bayesian method determines the best system state vector estimate, $\hat{\mathbf{x}}_{MMAE}^{Bayesian}(t_i^+)$, using a weighted *blending* of the elemental filters' system state vector estimates. The appropriate weighting placed during blending is the conditional probability associated with each elemental filter. The Bayesian-computed best system state vector estimate can be

expressed as

$$\hat{\mathbf{x}}^{Bayesian}_{MMAE}(t_i^+) = \mathbf{E}\left\{\hat{\mathbf{x}}(t_i^+)|\mathcal{Z}(t_i) = \mathcal{Z}_i\right\} = \sum_{k=1}^{K} \hat{\mathbf{x}}_k(t_i^+)p_k(t_i) \tag{2.24}$$

and the Bayesian-computed best parameter value estimate can be expressed as

$$\hat{\mathbf{a}}^{Bayesian}_{MMAE}(t_i) = \mathbf{E}\left\{\mathbf{a}(t_i)|\mathcal{Z}(t_i) = \mathcal{Z}_i\right\} = \sum_{k=1}^{K} \mathbf{a}_k(t_i)p_k(t_i) \tag{2.25}$$

The Bayesian-computed state estimate is shown in Figure 1. A Bayesian-computed parameter value could be depicted by replacing the product summation "$\hat{\mathbf{x}}_k p_k$" with the production summation "$\mathbf{a}_k p_k$", where $\mathbf{a}_k(t_i)$ corresponds to each elemental filter's hypothesis.

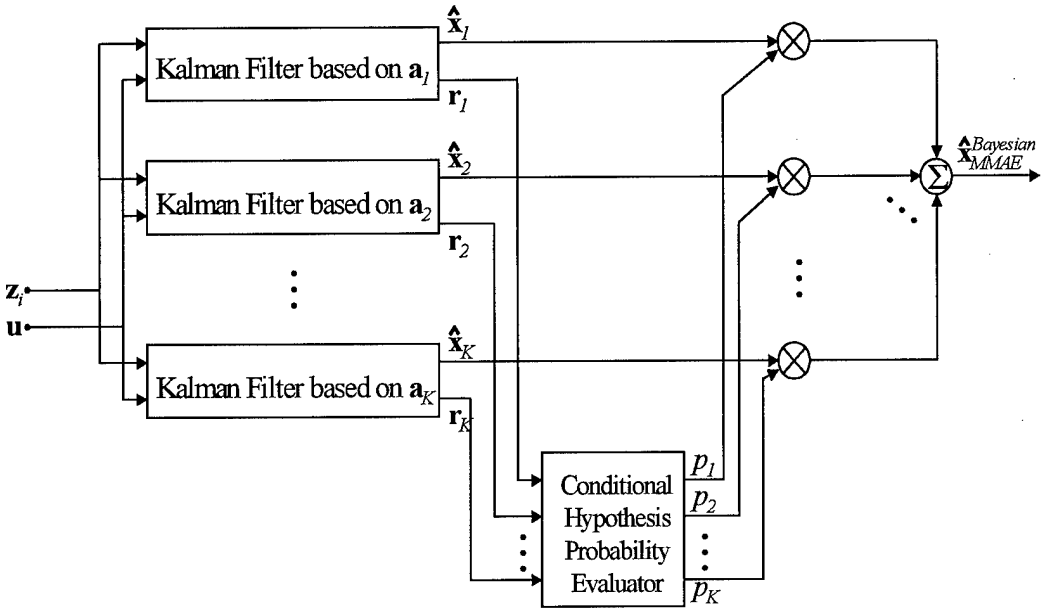


Figure 1. MMAE Bayesian State Estimate Block Diagram

Second, the maximum a posteriori (MAP) method uses the conditional *mode* (or value locating the maximum of the associated probability density) as its *best* estimates of the system parameter vector and system state vector. MAP determines the best system parameter vector estimate, $\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i)$, and the best system state vector estimate, $\hat{\mathbf{x}}^{MAP}_{MMAE}(t_i^+)$, as the parameter value hypothesized and state estimate given by the single elemental filter with the highest conditional probability.

This MAP-computed best system state vector estimate can be expressed as

$$\hat{\mathbf{x}}_{MMAE}^{MAP}(t_i^+) = \hat{\mathbf{x}}_j(t_i^+) \quad \text{where} \quad j = \arg\left\{\max_k [p_k(t_i)]\right\} \tag{2.26}$$

where $\arg\{\max[\cdot]\}$ is the argument of the maximum being performed. Similarly, the MAP-computed best parameter vector estimate can be expressed as

$$\hat{\mathbf{a}}_{MMAE}^{MAP}(t_i) = \mathbf{a}_j(t_i) \quad \text{where} \quad j = \arg\left\{\max_k [p_k(t_i)]\right\} \tag{2.27}$$

The MAP-computed state estimate could be depicted using the block diagram in Figure 1 with the product summation "$\hat{\mathbf{x}}_k p_k$" replaced with the production summation."$\hat{\mathbf{x}}_k p_k'$", where

$$\begin{aligned} j &= \arg\left\{\max_k [p_k(t_i)]\right\} \\ p_k' &= 0 \quad \text{for} \quad k \neq j \\ &= 1 \quad \text{for} \quad k = j \end{aligned} \tag{2.28}$$

An MAP-computed parameter value could be depicted by replacing the product summation "$\hat{\mathbf{x}}_k p_k$" with the production summation "$\mathbf{a}_k p_k'$" , where p_k' was defined in Equation (2.28) and $\mathbf{a}_k(t_i)$ corresponds to each elemental filter's hypothesis.

Notice that $\hat{\mathbf{a}}_{MMAE}^{MAP}(t_i)$ can only be one of the hypothesized values for the system parameter $\mathbf{a}$. The Bayesian method is appealing because the resultant $\hat{\mathbf{a}}_{MMAE}^{Bayesian}(t_i)$ need not be a hypothesized parameter vector. Admittedly, the true parameter vector will not likely fall on a hypothesized parameter vector; thus, the Bayesian method attempts to provide a better parameter estimate than MAP when this is the case. For this research effort, the Bayesian approach will be taken.

2.3.2.4 ***Review.*** It is important to review what information MMAE provides. For each filter and its assumed parameter value $\mathbf{a}_k$, MMAE provides a state estimate, $\hat{\mathbf{x}}_k(t_i^+)$, the filter-computed covariance, $\mathbf{A}_k$, and the MMAE-computed conditional probability, $p_k(t_i)$. Further, using conditional probability values, MMAE determines the *best*[5] estimate of the system parameter vector, $\hat{\mathbf{a}}_{MMAE}(t_i^+)$, and system state vector, $\hat{\mathbf{x}}_{MMAE}(t_i^+)$.

[5]Recall this is done through either the MAP or Bayesian approach.

2.3.3 *Control*

In most applications, the problem is not complete with just system parameter and state vector estimation. Often times, the main goal is to apply the appropriate control to the system. An appropriate control vector can be extracted from the multiple model structure. While there are several ways to form the appropriate control vector within a multiple model framework, two are discussed here: Multiple Model Adaptive Control and MMAE-based control.

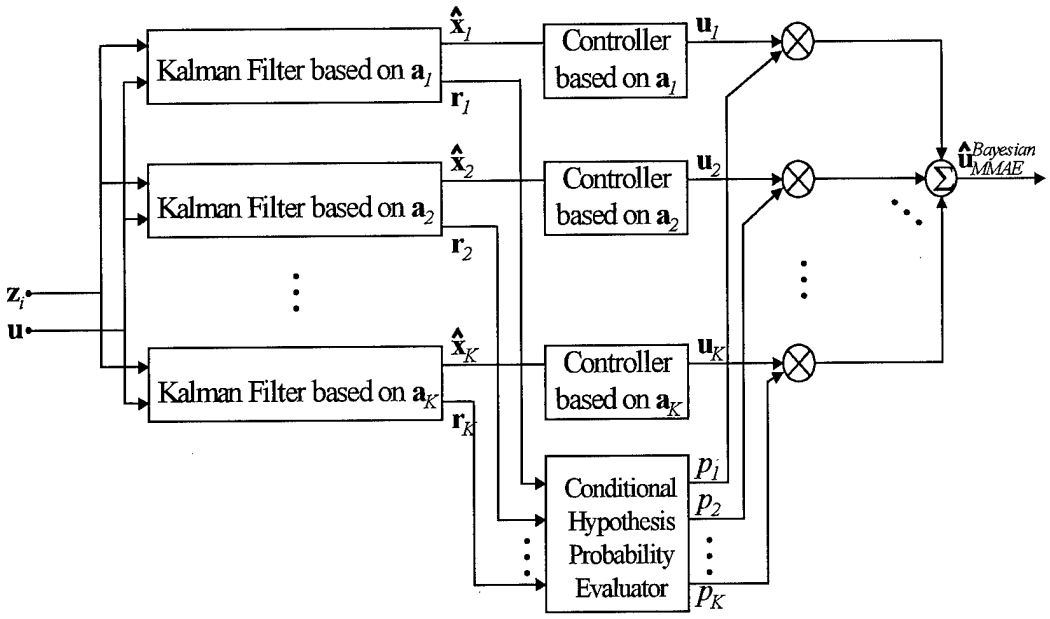


Figure 2. Multiple Model Adaptive Control Block Diagram

The first method is Multiple Model Adaptive Control (MMAC), shown in Figure 2. In MMAC, appropriate controllers are design for each hypothesized $\mathbf{a}_k$. The input to each controller is the system state vector estimate from the elemental Kalman filter associated with that controller, $\hat{\mathbf{x}}_k\left(t_i^+\right)$. The output of each controller is the appropriate control vector, $\mathbf{u}_k\left(t_i\right)$, based solely on the hypothesized system parameter vector, $\mathbf{a}_k$, given the input $\hat{\mathbf{x}}_k\left(t_i^+\right)$. Then, just as MMAE determines the

best[6] $\hat{\mathbf{a}}_{MMAE}(t_i)$ or $\hat{\mathbf{x}}_{MMAE}(t_i^+)$, MMAC determines the *best* control vector, $\mathbf{u}_{MMAC}(t_i)$. Therefore, $\mathbf{u}_{MMAC}(t_i)$ may consist of a weighted blend of all the control vectors given by the elemental controllers, denoted $\mathbf{u}_{MMAC}^{Bayesian}(t_i)$, where the weighting is the conditional probability associated with each elemental filter (as shown in Figure 2), or it may consist of only the control vector from the elemental controller that corresponds to the filter with the highest conditional probability, denoted $\mathbf{u}_{MMAC}^{MAP}(t_i)$, (by redefining the p_k's in Figure 2 as discussed previously in Equation (2.28)).

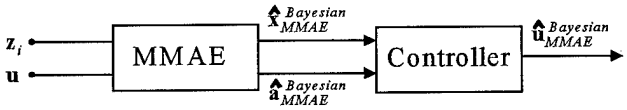


Figure 3. Multiple Model Adaptive Estimation Based Control Block Diagram

The second method is MMAE-based control, shown in Figure 3. The "MMAE" block represents the block diagram shown in Figure 1. In MMAE-based control, one controller is used to determine the control vector. MMAE is used to determine the best estimate of the system parameter vector, $\hat{\mathbf{a}}_{MMAE}(t_i)$, and the system state vector, $\hat{\mathbf{x}}_{MMAE}\left(t_i^+\right)$. The controller provides the control vector, $\mathbf{u}_{MMAE}(t_i)$, based (possibly) on the system parameter vector $\hat{\mathbf{a}}_{MMAE}(t_i)$, given the input $\hat{\mathbf{x}}_{MMAE}\left(t_i^+\right)$. If the system state and parameter vector estimates were obtained through a Bayesian approach, the control vector could be called $\mathbf{u}_{MMAE}^{Bayesian}(t_i)$ (as shown in Figure 2). Likewise, if the system state and parameter vector estimates were obtained through a MAP approach, the control vector could be called $\mathbf{u}_{MMAE}^{MAP}(t_i)$ (by redefining the p_k's used in the "MMAE" block in Figure 2 as discussed previously).

Comparing these two methods, MMAC should provide the best results because each controller is tailored for a particular $\mathbf{a}_k$. The disadvantage is that more memory storage may be required to include many controllers and, in applications in which controllers already exist, designing new, untested controllers may be unfeasible. Designing new controllers may be avoided if MMAE-

[6]Recall this is done through either the MAP or Bayesian approach.

based control is used. Further, the designer could provide information about the system (through $\hat{\mathbf{a}}_{MMAE}(t_i)$) to the controller. The controller could then be *adaptive*, changing dependent upon the estimate of the parameter vector.

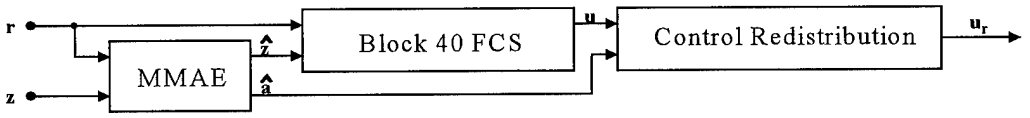


Figure 4. Multiple Model Adaptive Estimation with Control Redistribution

For this research, Control Redistribution, a specific implementation of MMAE-based control using a non-linear post-processor, is used to redirect the control authority as shown in Figure 4. (Section 3.6 details the development, implementation, and modification of CR.) The single validated controller, the VISTA F-16 Block 40 controller, is used to generate control commands, but then those command outputs are redistributed to appropriate actuators, depending on the estimate of the failure status parameter.

2.3.4 Hierarchical Structure

Recall that Multiple Model Adaptive Estimation (MMAE) is based on a *set* of system parameter vector values. For instance, the MMAE in Figure 1 is based on the set

$$\mathcal{A}=\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_K\}$$

In the case in which the parameter vector represents the failure status of an aircraft, if only single, complete failures were considered, this would require $K = 12$ (one fully functional filter and 11 filters corresponding to the 5 actuator failures and 6 sensor failures of interest). If single and dual failures were considered, this would require

$$K = 1 + 11 + \frac{(11)!}{(11-2)!\,(2)!} = 1 + 11 + 11\,(5) = 67$$

where K includes one fully functional filter, 11 single-failure-only filters corresponding to the 11 failures of interest, and 55 additional dual-failure combinations (taken two at a time in which order does not matter). The resulting MMAE bank would use $K = 67$ filters! Each filter must be

propagated and updated to obtain a state estimate, and the conditional probabilities for each filter must be calculated to be used in forming a state estimate and parameter vector estimate. It can be seen that using one MMAE bank to handle single and double failures simultaneously can become overwhelming quickly.

Although parallel computing greatly reduces the computational load of the filter equations [12, 17], the use of a hierarchical structure greatly reduces *the number* of filters actually used simultaneously at any one time [12]. To do this, the hierarchical structure breaks $\mathcal{A}$ into subsets, where each subset is used in an MMAE. For instance, re-consider single and dual failures on an aircraft, where the parameter vector represents the failure status. Previously, one MMAE algorithm considered 67 hypotheses. The hypothesis set $\mathcal{A}$ was

$$\mathcal{A} = \left\{ \begin{array}{c} \mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_K, \\ \mathbf{a}_{12}, \mathbf{a}_{13}, \cdots, \mathbf{a}_{1K}, \\ \mathbf{a}_{21}, \mathbf{a}_{23}, \mathbf{a}_{24}, \cdots, \mathbf{a}_{2K}, \\ \vdots \\ \mathbf{a}_{K1}, \mathbf{a}_{K2}, \cdots, \mathbf{a}_{K(K-1)} \end{array} \right\}$$

where K is the number of failures. Note that $\mathbf{a}_{kj} = \mathbf{a}_{jk}$ because the order does not matter, that is, the failure model of an aircraft with two failures does not depend on the order in which the failures occurred. The hypothesis $\mathbf{a}_0$ is the fully functional hypothesis (i.e., no failure). The hypotheses with one index (i.e., $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_K$) correspond to the 11 single-failure-only hypotheses. The hypotheses with two indices (for example, $\mathbf{a}_{12}$) correspond to the dual failure combinations. The set $\mathcal{A}$ could be broken into the $K+1$ subsets (or *banks*, $\mathcal{B}$) of $(K+1)$hypotheses each:

$$\mathcal{A} = \{\mathcal{B}_0, \mathcal{B}_1, \cdots, \mathcal{B}_K\}$$

where each bank is defined

$$\begin{aligned} \mathcal{B}_0 &= \{\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_K\} \\ \mathcal{B}_1 &= \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_{12}, \cdots, \mathbf{a}_{1K}\} \\ \mathcal{B}_2 &= \{\mathbf{a}_0, \mathbf{a}_{21}, \mathbf{a}_2, \cdots, \mathbf{a}_{2K}\} \\ &\vdots \\ \mathcal{B}_K &= \{\mathbf{a}_0, \mathbf{a}_{K1}, \mathbf{a}_{K2}, \cdots, \mathbf{a}_K\} \end{aligned}$$

The banks are shown graphically in Figure 5. The bank $\mathcal{B}_0$ is considered "Level 0" since all hypotheses are single failures. The banks $\mathcal{B}_1$ through $\mathcal{B}_K$ are considered "Level 1" because they primarily consist of dual-failure hypotheses. Also notice that each "Level 1" bank has a no-failure hypothesis and a single-failure hypothesis. The single-failure hypothesis continues the declaration mode in "Level 0" about a specific single failure having occurred, while the dual-failure hypotheses continue that original failure declaration plus allow for any additional failure. Each bank can be used as the basis for an MMAE (as shown in Figure 1). For example, the MMAE may start "looking" for single failures only, using bank $\mathcal{B}_0$ in "Level 0". If a single failure occurs, then the MMAE uses a new bank in "Level 1" to look for dual failures.

The "arrows" in Figure 5 show the new bank in "Level 1" that corresponds to the failure hypothesis in "Level 0". Further, inside each arrow is the hypothesis that causes a bank change. The no-failure hypothesis is included in each "Level 1" bank in case the declaration that caused the use of a new bank was incorrect. The single-failure hypothesis that caused the use of a new bank is included in the "Level 1" bank in case there is only a single failure. Thus, when considering single and dual failures, the number of on-line filters used simultaneously at any one time is always $K + 1 = 12$. The only additional load to consider dual failures rather than single failures is the switching logic!

2.4 MMAE in Practice

In engineering, it is essential to *bridge the gap* between theory and practice: *"In theory, there is no difference between theory and practice. In practice, there may very well be a substantial difference between theory and practice."* Multiple Model Adaptive Estimation (MMAE) presented thus far is complete and theoretically sound, but many practical enhancements can be made before implementation. To *bridge this gap*, this section will present topics fashioned specific to this research, including particular values chosen where appropriate.

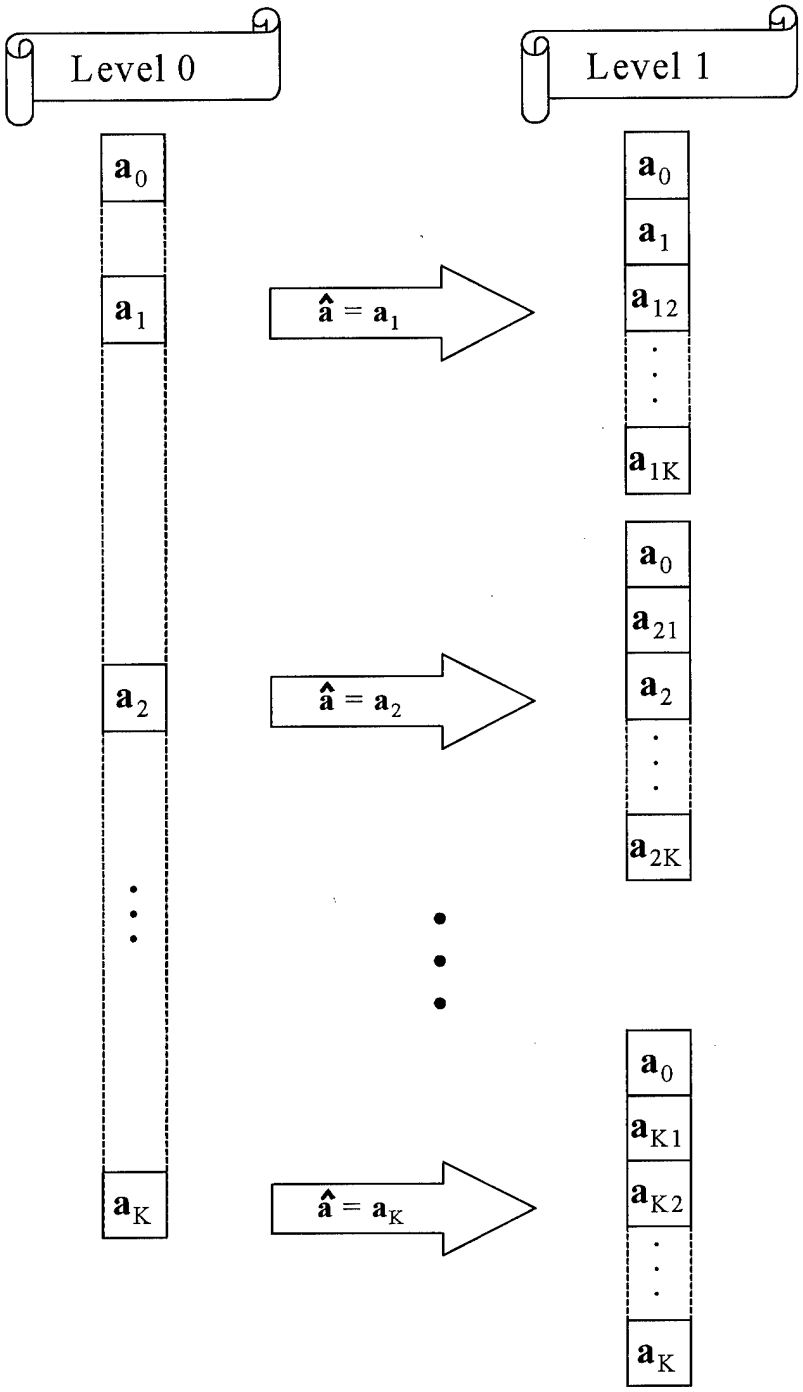


Figure 5. Hierarchical Structure Set Definition

2.4.1 Lower Bounding

2.4.1.1 Probability Lower Bound. Due to the recursive nature of the probability calculation given in Equation (2.19), if a probability ever becomes zero, it will remain zero thereafter and the system will become insensitive to change. To prevent this from occurring, the probabilities are *artificially* lifted off zero by introducing a probability lower bound [1,23]. Because the probabilities build over time (again, due to the recursive nature of Equation (2.19)), applying this probability lower bound improves system response. For this research effort, the lower bound was selected to be 0.001 [28, 43]. Larger values force inappropriately large weights to be associated with parameter values that are actually incorrect, while smaller values yield too sluggish a response in computed probabilities when a true parameter change occurs.

An alternate approach to a probability lower bound is to use a Markov model to represent parameter variations over time. A state transition matrix progresses a vector of $p_k(t_i)$ probabilities from time step to time step. The advantage is the probability lower bound is not needed. The disadvantage is the full state transition matrix is usually not known. The designer must employ ad hoc methods to fill the state transition matrix. Often times, as with this research, it is easier to select probability lower bounds that make sense than it is to fill the state transition matrix.

2.4.1.2 Blending Lower Bound. Placing a lower bound on the probabilities after Equation (2.19) is used to compute the $p_k(t_i)$ values ensures that, no matter how *wrong* a filter's estimate is, it will have a probability of at least the lower bound. The Bayesian method as presented uses *all* filters to provide state vector and parameter vector estimates. Though their associated probabilities may be small, filters based on *wrong* parameter values can only degrade the MMAE estimate. To alleviate this problem, the *modified* Bayesian approach applies a blending lower bound as well as a probability lower bound. If a conditional probability falls below some threshold, it is not used in the blending algorithm used to generate $\hat{\mathbf{x}}_{MMAE}(t_i)$ or $\hat{\mathbf{a}}_{MMAE}(t_i)$. By so doing, blending of state estimates and parameter values from filters with reasonably large $p_k(t_i)$'s is enabled, as desired, but

erroneous filters' values (values from filters with $p_k(t_i)$'s artificially bounded by minimum values) do not corrupt $\hat{\mathbf{x}}_{MMAE}(t_i)$ or $\hat{\mathbf{a}}_{MMAE}(t_i)$. A blending lower bound of 0.003 will be used for this research [10,11].

2.4.1.3 Summary. There are two lower bounds used in this research, a blending lower bound of 0.003 and a probability lower bound of 0.001. If the probability or blending lower bound is used, then the probabilities must be re-scaled to sum to one. This research effort incorporates such logic [8,28,43].

2.4.2 Beta Dominance

Beta dominance[7] [1] is the tendency of the probability evaluator to calculate probabilities incorrectly because of an erroneous contribution by the β_k term of Equations (2.21) and (2.22). The result is a tendency to declare incorrect failures corresponding to filters with the smallest precomputed residual covariance, $\mathbf{A}_k$. Consider the situation where two filters have nearly the same value for the likelihood quotient, $L_1 = L_2 = L$. One would expect that the MMAE would assign equal probabilities to both filters, and indeed the exponential term of the probability density, Equation (2.21), will have the same value, $e^{-\frac{1}{2}L}$. However, the β_k term will typically differ, based on the magnitude of the covariance, $|\mathbf{A}_k|$, resulting in the filter with the smaller (in the norm sense) filter-predicted covariance being weighted more highly than the other. Recall from Equation (2.6) that the residual covariance is calculated as $\mathbf{A}_k(t_i) = \mathbf{H}_k\mathbf{P}_k(t_i^-)\mathbf{H}_k{}^T + \mathbf{R}_k$, and that its value has *nothing* to do with the correctness or incorrectness of a hypothesized parameter value. In this research effort, the $\mathbf{H}$ matrix for some filters (those associated with sensor failures) will have a row of zeros (see Section 3.5). All other things equal, the filter with the row of zeros will have a smaller $\mathbf{A}_k$ and will be given higher probability. Research has shown this effect [28,43].

Two methods have been used to remove the beta dominance effect. The first uses scalar residual monitoring [43], discussed in Section 2.4.6, to provide additional votes to determine whether a declaration is correct. While demonstrated effective at eliminating false alarms (on sensor fail-

[7]This section is heavily based upon Stepaniak's thesis [41].

ures) [28, 31, 32, 43], this method does not prevent beta dominance from adversely affecting the MMAE by introducing erroneously high probabilities. A second technique simply removes the β_k term altogether, i.e., equates all β_k's to one, in Equation (2.21) [29, 43]. This technique has demonstrated even better results than that of residual monitoring [10, 11, 27], and will be used in this research. Even though the resulting expression

$$f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1}) = e^{[-\frac{1}{2}L_k(t_i)]} \tag{2.29}$$

is no longer a proper Gaussian density (the area under the function is not unity), this is of no overriding concern since the density function is used only to calculate the p_k probabilities in Equation (2.19). The normalizing effect of the denominator in Equation (2.19) assures that use of Equation (2.29) still yields a valid set of probabilities in the sense that Equation (2.20) is satisfied.

2.4.3 Scalar Penalty

The scalar coefficient[8] of $-\frac{1}{2}$ in the exponential term of Equation (2.21) is directly related to the sensitivity of the MMAE [27]. Increasing the magnitude of this term drives probabilities associated with large residuals to zero faster. Ideally, this should result in a faster adaptation process; however, increasing the scalar penalty also increases the incidence of false alarms. The $-\frac{1}{2}$ coefficient can then be treated as a sensitivity gain which can be adjusted to tune performance. For this research, the scalar coefficient is maintained at the original value of $-\frac{1}{2}$.

2.4.4 Dither

Dither[9] is the introduction of low magnitude control signals in order to excite the system and enhance identifiability. To illustrate the need for a dither signal, consider an aircraft flying in a straight and level flight path. Should a rudder fail, the loss or degradation in control authority may well go unnoticed by both the pilot and the adaptive controller, particularly if the surface fails in such a manner that it floats in the relative wind (a failure to free stream). Only after the pilot initiates a maneuver will the residuals in the MMAE begin to reflect the change in system parameters, resulting

[8]This section is heavily based upon Stepaniak's thesis [41].

[9]This section is heavily based upon Stepaniak's thesis [41].

in a period of degraded performance until the failure is declared. A better design detects the failure when it happens regardless of flight path, allowing the MMAE to reconfigure appropriately and provide maximum performance to the pilot at all times. Subliminal dither signals force this early detection by exercising the control surfaces, which excites the state vector. Any deviation from the filter-predicted state values will then show up in the residuals and trigger a failure declaration.

Though several forms of dither have been suggested [31, 32], varying waveform, frequency content, and amplitude, arguably the best for enhancing failure detection is a simple sinusoid since a single sinusoid at a known frequency is so simple to detect in a signal (the appearance of the dither signal in the filter residual is indicative of the filter's assumed parameter value being incorrect [27, 31,32]). A periodic dither has the additional advantage that sophisticated algorithms can monitor the residuals not just for magnitude but also for frequency effects corresponding to the input dither. By sending a different frequency or phase to each channel, the confounding effects of cross-coupling between channels can be minimized.

In applications in which humans are involved, as in this research, dither signals need to be subliminal. An associated drawback is the smaller amplitudes required for subliminal dither are less effective at enhancing parameter identification. An alternative implementation may be to introduce larger magnitudes, but infrequently and only for short periods of time, possibly even allowing the human to control their application. Note that dither is only needed during benign conditions when the MMAE does not receive enough information to make accurate decisions. During strong maneuvers, the system is excited sufficiently without the addition of dithering. In fact, dither may not be needed at all if humans were trained to induce maneuvers which shook up the system either periodically throughout the flight or just before the onset of deliberate maneuvers.

The dither signal used in this research is chosen based on previous research [18,41,42] to be sinusoidal with a frequency of 15 Hz. The effects of vibration on human subjects was researched in the 1960's in conjunction with the space program [13,38]. Although scientists at the time were considering unwanted vibrations, their conclusions are equally pertinent to dither, which can be

thought of as an induced vibration. Based on their findings, the limits on maximum allowable dither magnitude response used for this research is set to be $\pm$ 0.1 g's in the longitudinal channel and $\pm$ 0.2 g's in the lateral channel. The signal applied is not considered subliminal; rather, it should be thought of as *tolerable* to the pilot.

2.4.5 Probability Smoothing and Quantization

As the conditional probabilities progress through time, they may exhibit transients or jagged behavior before converging to the correct solution. Probability smoothing [27] is introduced to reduce the possibility that the MMAE will declare false alarms based solely on transients. Removing these false alarms becomes even more important when using a hierarchical structure since an alarm declaration results in the onset of a new bank of MMAE filters. The probabilities are smoothed by averaging over a moving window of a given size, which is specific to each application. Probability quantization may be used to force the probabilities to discrete values. This may be useful in making decisions and in applying blending techniques [8]. For this research, no probability smoothing (effectively, a window size of one) or probability quantization is used.

2.4.6 Scalar Residual Monitoring

The MMAE calculates probabilities based on the likelihood quotient, $L_k(t_i)$. Additional insight can be gained by monitoring the scalar residuals themselves[10] [28,31,32,43]. Some hypotheses[11] may appear not only as an increase in the likelihood quotient, but also as a direct increase in one of the corresponding scalar residuals. Checking these terms can then be used as an additional vote either to increase the convergence time if there is an indication of convergence to the wrong hypothesis, or to eliminate false alarms. For example, when beta dominance causes false alarms due to a particular filter, the MMAE algorithm can test the scalar residual corresponding to that filter. If the result of the test does not corroborate the declaration, then the declaration is labeled false and is ignored.

[10]This section is heavily based upon Stepaniak's thesis [41].

[11]Past research has shown that a failed sensor is manifested in such a manner.

Several methods exist for testing scalar residuals, all of which rely on the statistical properties of residuals, which are white Gaussian processes with zero mean and a variance of the appropriate diagonal term of $\mathbf{A}_k(t_i)$, provided that the presumed k^{th} hypothesis is correct. A simple way to evaluate the whiteness of the residual is to count the number of zero crossings [31, 32], once the residuals are corrected for zero-meanness. A white process is indicated by a significantly higher count than would be present for colored noise. In practice, an arbitrary threshold can be set, above which whiteness is declared satisfied. Another test for correct variance versus whiteness counts the number of times that the residual breaks the $\pm\sigma$ boundaries [23]. For a true Gaussian distribution, the residual should be inside the one-σ boundary 68.3% of the time. Similarly, statistics for the two- and three-σ boundaries are 95.4% and 99.7%, respectively. A third test is to calculate the mean of a finite number of samples. Any deviation from a small range about zero could then indicate, or be used to corroborate, a declaration.

Other more sophisticated tests also exist, such as calculating the scalar quadratic term $r_{kj}^2/A_{k_{jj}}$, representing the ratio of the true square of the j^{th} scalar residual to its filter predicted variance in the k^{th} elemental filter [31,32]. Similar to the likelihood quotient calculation, filters with the correct hypothesis will have residual-squared values most in consonance with the filter's predicted variance. This method is particularly useful for detecting hypotheses which tend to affect a single scalar residual, rather than hypotheses which appear in multiple scalar residuals (and are thus harder to detect in general for any method). Also, if dither in the form of a sinusoid at a known frequency is applied, the appearance of that dither frequency in a given residual is another indication of incorrectness of the corresponding hypothesis [27, 31, 32]. The dither, which is very nonwhite and easily detected since it is of known frequency, should not appear in the otherwise zero-mean, white residuals if the corresponding hypothesis is correct. In this research, scalar residual monitoring will not be used.

2.5 Chapter Summary

This chapter was meant to provide a basic overview of MMAE in general. The history of MMAE was given, from its early development to its current use on the VISTA F-16. The building blocks of MMAE were reviewed. A Kalman filter based upon one system parameter vector realization provides a single system state vector estimate. Putting multiple Kalman filters together based upon different system parameter vector realizations, a MMAE can provide a better state vector estimate than a single Kalman filter *and* provides a system parameter vector estimate. Putting multiple MMAE banks together, a hierarchical structure can cover a larger set of system parameter vector realizations while minimizing the number of on-line filters. To go from these theoretical building blocks to practice, we *bridge the gap* by applying practical enhancements such as probability lower bounds, blending lower bounds, beta dominance alleviation techniques, dither, gain adjustments of scalar penalties, probability smoothing, probability quantization, and scalar residual monitoring.

Chapter 3 - Problem Definition and Design Model

3.1 Chapter Overview

This chapter provides the models used for this research effort, stepping the reader from the "real world" to the truth model to the design model. The "real world" VISTA F-16 is introduced in Section 3.2. The truth model, a high-fidelity representation of the "real world" VISTA F-16, is presented in Section 3.3. A design model (of reduced-order compared to the truth model) is presented in Section 3.4. Modifications to the truth model and design model to incorporate failures are presented in Section 3.5. Control Redistribution is presented and modified in Section 3.6.

3.2 "Real World"

This research is performed on the Variable-Stability In-flight Simulator Test Aircraft (VISTA) F-16. This aircraft was developed to provide an in-flight simulator for various high performance military aircraft. The control system of the VISTA F-16 can be configured so that the aircraft will emulate the characteristics of other aircraft. This capability provides an extremely flexible tool to develop and evaluate aircraft control systems. This capability also provides a safe and realistic means to train pilots for these systems. The VISTA F-16 was built using a modified F-16D airframe by Calpsan and General Dynamics [33].

The principle reason for using the VISTA F-16 in this research is to facilitate the continuation of previous research which has been performed at AFIT [8, 10, 11, 18, 31, 32, 41, 42]. Originally, the primary reason for choosing this platform was based on the availability of an accurate truth model and advanced simulation tools at the Air Vehicles Directorate, Air Force Research Laboratory, Wright Patterson Air Force Base, OH (formerly Wright Laboratory).

3.3 Truth Model

In order to evaluate the performance of the MMAE and Control Redistribution algorithm, a truth model is needed which will provide an *accurate simulation of the real world environment and a proper statistical portrayal of estimation errors committed by each filter and the MMAE in that "real*

world" environment [22]. As mentioned in Section 3.2, the truth model was provided by the Flight Dynamics Directorate, Air Force Research Laboratory, Wright Patterson Air Force Base, OH.

The VISTA simulation used for the truth model runs as part of the Simulation Rapid-prototyping Facility (SRF) and incorporates General Dynamics' VISTA F-16 simulation software with variable stability flight control system software provided by Calpsan. The SRF VISTA F-16 simulation provides a full six-degree-of-freedom truth model incorporating the aircraft's nonlinear equations of motion, advanced actuator modeling, the complete Block 40 controller, and the aileron-to-rudder interconnect used to provide coordinated turns. The Transportable Applications Executive (TAE) provides the interface to allow the user to configure the F-16, select a flight condition, and command the pilot inputs in non-real time.

Throughout this sequence of research, several modifications have been made to the FORTRAN source code of the SRF VISTA F-16 simulation [8,10,11,18,31,32,41,42]. These changes were made to accommodate the multiple model architecture and to provide a more accurate depiction of the real world environment. These changes include the incorporation of a more precise wind model based on a zero-order Dryden wind model [10, 11, 37], the incorporation of code which enhances failure simulation (allowing single or dual and partial or complete failures) [8, 10, 11, 18, 31, 32, 41, 42], the incorporation of sensor noise [10, 11], and the replacement of the lateral acceleration measurement computation by a more realistic linear model [41,42] (see Section 3.4.6).

3.4 Design Model

Having a truth model that accurately models the "real world", a design model must be found such that the MMAE and Control Redistribution algorithm based on it can be implemented on the 64Hz digital flight computer of the VISTA F-16, while capturing the important characteristics of the truth model. The design model is selected as a linear, time-invariant, discrete-time model, of reduced order compared to the truth model. The truth model can be of higher fidelity because it is not implemented on the digital flight computer, rather it is simply used to *verify* the design.

To obtain the design model, the non-linear, time-variant, continuous-time model is *linearized* about a nominal aircraft configuration and flight condition, providing a linear, time-invariant model, as discussed in Section 3.4.1. In Section 3.4.2, the input vector and input matrix are redefined in order to model failures and apply Control Redistribution. In Section 3.4.3, a design model of the actuators is developed. In Section 3.4.4, the actuator model is incorporated into the system model. In Section 3.4.5, the *continuous-time* model is converted to an *equivalent, discrete-time* model. In Section 3.4.6, a *discrete-time* measurement model is properly defined. Section 3.4.7 will review the discrete-time, linear, time-invariant design model.

3.4.1 Linearization About a Nominal Aircraft Configuration and Flight Condition

In order to provide a linear, time-invariant design model, the truth model must be linearized about a nominal aircraft configuration and flight condition. The TAE provides three aircraft configurations, defined in Table 1. To be consistent with previous research, the aircraft configuration for this research is selected as "up-and-away". This selection configures the aircraft with the leading edge flaps up and the landing gear up, a typical configuration for the steady, level flight assumed. The TAE also accommodates flight conditions within the flight envelope of the VISTA F-16. To be consistent with previous research, the flight condition for this research is a speed of Mach 0.4 and an altitude of 20,000 feet. This flight condition was chosen because it results in a low dynamic pressure, causing failure detection to be more difficult[12]. Thus, the intent is to demonstrate algorithm capabilities in the most challenging environment for failure detection, rather than under the most favorable conditions.

Table 1. Aircraft Configurations

Aircraft Configuration	Leading Edge Flaps	Landing Gear
Takeoff	Up	Down
Up-and-away	Up	Up
Landing	Down	Down

[12]It is assumed that, in low dynamic pressure, actuators have less control authority than in high dynamic pressure. Therefore, the effects of a failed actuator may be more difficult detect.

With the aircraft configured and the flight condition chosen, a linear, time-invariant (LTI) design model can be formed as

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{3.1}$$

where $\mathbf{x}(t)$ is the state vector, $\mathbf{A}$ is the plant matrix, $\mathbf{u}(t)$ is the control vector, $\mathbf{B}$ is the input matrix, $\mathbf{w}(t)$ is the dynamics driving noise, and $\mathbf{G}$ is the noise injection matrix.

3.4.1.1 State Vector. The state vector defined in Equation (3.1) is described in Table 2. It is grouped in longitudinal components $\{\theta, u, \alpha, q\}$ and lateral components $\{\phi, \beta, p, r\}$. These channels are assumed independent, as the form of $\mathbf{A}$ in Equation (3.2) in the next paragraph indicates.

Table 2. State Vector Description

State Vector
$\mathbf{x}(t) = \begin{bmatrix} \theta(t) \\ u(t) \\ \alpha(t) \\ q(t) \\ \phi(t) \\ \beta(t) \\ p(t) \\ r(t) \end{bmatrix}$

Symbol	Description	Units
θ	Pitch Angle	rad
u	Forward Velocity	ft/sec
α	Angle of Attack	rad
q	Pitch Rate	rad/sec
ϕ	Bank Angle	rad
β	Sideslip Angle	rad
p	Roll Rate	rad/sec
r	Yaw Rate	rad/sec

3.4.1.2 Plant Matrix. The plant matrix, $\mathbf{A}$, used in Equation (3.1) is defined as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ X'_\theta & X'_u & X'_\alpha & X'_q & 0 & 0 & 0 & 0 \\ Z'_\theta & Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 & 0 \\ M'_\theta & M'_u & M'_\alpha & M'_q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \phi'_r \\ 0 & 0 & 0 & 0 & Y'_\phi & Y'_\beta & Y'_p & Y'_r \\ 0 & 0 & 0 & 0 & 0 & L'_\beta & L'_p & L'_r \\ 0 & 0 & 0 & 0 & 0 & N'_\beta & N'_p & N'_r \end{bmatrix} \tag{3.2}$$

Notice the longitudinal derivatives in the upper left are augmented with the lateral derivatives in the lower right. The variables X', Y', and Z' represent the aerodynamic *forces* in the x, y, and z aircraft body frame axes, respectively. The aircraft body frame is defined with the positive x-axis out the nose of the aircraft, the positive y-axis out the right wing of the aircraft, and the positive

z-axis in the down direction to complete a right-hand reference frame, where the origin is arbitrarily placed in the body of the aircraft. (The origin is fixed during the design stages of the airframe and does *not* coincide with the center of gravity. The center of gravity varies with payload. The vector from the origin to the center of gravity is defined as $x_{cg}\hat{\imath} + z_{cg}\hat{\mathbf{k}}$ where $\hat{\imath}$ and $\hat{\mathbf{k}}$ are unit vectors in the x and z axes directions, respectively, and the difference in the y-axis direction is usually neglected.) The variables L', M', and N' represent the *moments* about the x, y, and z axes, respectively. The primed notation, $'$, indicates that the variables are in terms of the angle of attack, α, and sideslip angle, β, rather than the velocities v and w in the y and z axes directions respectively. The variable subscript is a shorthand notation indicating a partial derivative (for example, $X'_\theta \triangleq \frac{\partial X'}{\partial \theta}$). The subscript symbols correspond to the states defined in Table 2. For example, X'_θ could be read, "the partial derivative of the aerodynamic force in the x-axis direction with respect to the pitch angle," or more descriptively, "the change in the aerodynamic force in the x-axis direction due to a change in the pitch angle."

3.4.1.3 Input Vector. The input vector defined in Equation (3.1) is described in Table 3.

Table 3. Input Vector Description

Input Vector
$\mathbf{u}(t) = \begin{bmatrix} \delta_e(t) \\ \delta_{dt}(t) \\ \delta_f(t) \\ \delta_a(t) \\ \delta_r(t) \end{bmatrix}$

Symbol	Description	Units
δ_e	Elevator Position	rad
δ_{dt}	Differential Tail Position	rad
δ_f	Flap Position	rad
δ_a	Aileron Position	rad
δ_r	Rudder Position	rad

3.4.1.4 Input Matrix. The input matrix, $\mathbf{B}$, used in Equation (3.1) is defined as

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ X'_{\delta_e} & 0 & X'_{\delta_f} & 0 & 0 \\ Z'_{\delta_e} & 0 & Z'_{\delta_f} & 0 & 0 \\ M'_{\delta_e} & 0 & M'_{\delta_f} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & Y'_{\delta_{dt}} & 0 & Y'_{\delta_a} & Y'_{\delta_r} \\ 0 & L'_{\delta_{dt}} & 0 & L'_{\delta_a} & L'_{\delta_r} \\ 0 & N'_{\delta_{dt}} & 0 & N'_{\delta_a} & N'_{\delta_r} \end{bmatrix} \tag{3.3}$$

Again, the variables X', Y', and Z' represent the aerodynamic *forces* in the x, y, and z axes, respectively, and the variables L', M', and N' represent the *moments* about the x, y, and z axes, respectively. As before, the primed notation, $'$, indicates that the variables are in terms of the angle of attack, α, and sideslip angle, β, and the variable subscript indicates a partial derivative. The subscript symbols correspond to the inputs defined in Table 3. For example, X'_{δ_e} could be read, "the partial derivative of the aerodynamic force in the x-axis direction with respect to the elevator position," or more descriptively, "the change in the aerodynamic force in the x-axis direction due to a change in the elevator position."

3.4.1.5 Dynamics Driving Noise. The dynamics driving noise vector, $\mathbf{w}(t)$, as defined in Equation (3.1) is considered white Gaussian noise, with a mean of zero and strength $\mathbf{Q}$ as given in Table 4. From the definition of $\mathbf{Q}$, it can be seen that $\mathbf{w}(t)$ is a $6x1$ vector with white Gaussian noise pertaining to the states $\{u, \alpha, q, p, \beta, r\}$. Further, notice that the white Gaussian noise for α is correlated with the white Gaussian noise for q and the white Gaussian noise for β is correlated with the white Gaussian noise for r. All other noises are assumed uncorrelated.

Table 4. Dynamics Driving Noise Description

Q	Average Noise Strength		Units
Q(1,1)	u	4.5×10^{-2}	$ft^2/rad \cdot sec$
Q(2,2)	α	3.0×10^{-6}	$rad \cdot sec$
Q(2,3)	α vs. q	1.1×10^{-8}	rad
Q(3,3)	q	1.5×10^{-6}	rad/sec
Q(4,4)	p	6.0×10^{-6}	rad/sec
Q(5,5)	β	3.0×10^{-6}	$rad \cdot sec$
Q(5,6)	β vs. r	6.3×10^{-9}	rad
Q(6,6)	r	2.4×10^{-6}	rad/sec

3.4.1.6 Noise Injection Matrix. The noise injection matrix, $\mathbf{G}$, defined in Equation (3.1) is described as

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ X'_u & X'_\alpha & X'_q & 0 & 0 & 0 \\ Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 \\ M'_u & M'_\alpha & M'_q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y'_p & Y'_\beta & Y'_r \\ 0 & 0 & 0 & L'_p & L'_\beta & L'_r \\ 0 & 0 & 0 & N'_p & N'_\beta & N'_r \end{bmatrix} \tag{3.4}$$

Notice that $\mathbf{G}$ is a $8x6$ matrix, where the eight rows correspond to the system states and the six columns correspond to the six white Gaussian noises described as $\mathbf{w}(t)$.

3.4.1.7 Evaluation of the Plant, Input, and Noise Injection Matrices. Because the design model is linearized about an aircraft configuration and flight condition, the stability derivatives are constant (thereby yielding a time-invariant design model). Thus, the $\mathbf{A}, \mathbf{B}$, and $\mathbf{G}$ matrices are time-invariant and can be filled with actual values for this design. These values can be found in Appendix A. The stability derivatives are *dimensional*; the interested reader can verify that the units are consistent throughout the equations.

Further, the open-loop eigenvalues of the system can be obtained by evaluating the plant matrix. It is worth noting that the plant matrix contains an *unstable* longitudinal eigenvalue, making this flight control problem even more challenging [8]. Appendix A details the eigenvalues for the open-loop plant.

3.4.2 Redefine the Input Vector and Input Matrix

3.4.2.1 Input Vector. The input vector defined in Table 3 is a conventional choice of surface positions. Stepaniak [41, 42] developed a clever, equivalent choice of surface positions to redefine the input vector in order to model failures more easily and apply Control Redistribution (see Section 3.6). Rather than defining the input vector in conventional terms, the input vector is redefined in *physical* terms. Recall the input vector was defined as

$$\mathbf{u}(t) = \begin{bmatrix} \delta_e(t) & \delta_{dt}(t) & \delta_f(t) & \delta_a(t) & \delta_r(t) \end{bmatrix}^T \tag{3.5}$$

where the components are elevator position, differential tail position, flaps position, aileron position, and rudder position, respectively. The physical control surfaces that perform these functions are the left stabilator, right stabilator, left flaperon, right flaperon, and rudder.

Table 5. Modified Input Vector Description

Input Vector
$\mathbf{u}_{mod}(t) = \begin{bmatrix} \delta_{ls}(t) \\ \delta_{rs}(t) \\ \delta_{lf}(t) \\ \delta_{rf}(t) \\ \delta_{r}(t) \end{bmatrix}$

Symbol	Description	Units
δ_{ls}	Left Stabilator Position	rad
δ_{rs}	Right Stabilator Position	rad
δ_{lf}	Left Flaperon Position	rad
δ_{rf}	Right Flaperon Position	rad
δ_{r}	Rudder Position	rad

The input vector in terms of the physical control surface positions, termed the modified input vector, is defined in Table 5. The *modified* input vector is repeated here as

$$\mathbf{u}_{mod}(t) = \begin{bmatrix} \delta_{ls}(t) & \delta_{rs}(t) & \delta_{lf}(t) & \delta_{rf}(t) & \delta_{r}(t) \end{bmatrix}^T \tag{3.6}$$

To illustrate the relationship between the conventional notation and the physical notation, consider the mathematical interpretation:

$$\begin{bmatrix} \delta_e(t) \\ \delta_{dt}(t) \\ \delta_f(t) \\ \delta_a(t) \\ \delta_r(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_{ls}(t) \\ \delta_{rs}(t) \\ \delta_{lf}(t) \\ \delta_{rf}(t) \\ \delta_{r}(t) \end{bmatrix} \tag{3.7}$$

The stabilators and flaperons can be controlled symmetrically or asymmetrically. The first row indicates *symmetric* application of the stabilators produces the conventional *pitching* action accomplished by an elevator. The second row indicates *asymmetric* application of the stabilators produces the conventional *rolling* action (with adverse yaw) accomplished by a differential tail. The third row indicates *symmetric* application of the flaperons produces the conventional *pitching* action accomplished by the flaps[13]. The fourth row indicates *asymmetric* application of the flaperons produces

[13]Note that the "flaps" mentioned here are used during flight for maneuvers. The "leading edge flaps" are used during landing or in slow maneuvers and are not considered in the linearized model. To include the "leading edge flaps" in a linearized model, the aircraft configuration must be selected as "landing", as defined in Table 3-1, *before* linearization.

the conventional *rolling* action (with adverse yaw) accomplished by the ailerons. The fifth row indicates that the function of the rudder remains unchanged.

3.4.2.2 Input Matrix. The input matrix, **B** of Equation (3.3), must be redefined to accommodate the change from **u** to $\mathbf{u}_{mod}$ as expressed in Equation (3.7). Notice that Equation (3.7) is of the form

$$\mathbf{u}(t) = \mathbf{T}\,\mathbf{u}_{mod}(t) \tag{3.8}$$

The expression $\mathbf{Bu}(t)$ found in Equation (3.1) must be maintained, so by substitution and manipulation,

$$\mathbf{Bu}(t) = \mathbf{B}(\mathbf{Tu}_{mod}(t)) = (\mathbf{BT})\,\mathbf{u}_{mod}(t) = \mathbf{B}_{mod}\mathbf{u}_{mod}(t) \tag{3.9}$$

Thus, the modified input matrix (based on physical control surfaces), $\mathbf{B}_{mod}$, is

$$\mathbf{B}_{mod} = \mathbf{BT} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}X'_{\delta_e} & \frac{1}{2}X'_{\delta_e} & \frac{1}{2}X'_{\delta_f} & \frac{1}{2}X'_{\delta_f} & 0 \\ \frac{1}{2}Z'_{\delta_e} & \frac{1}{2}Z'_{\delta_e} & \frac{1}{2}Z'_{\delta_f} & \frac{1}{2}Z'_{\delta_f} & 0 \\ \frac{1}{2}M'_{\delta_e} & \frac{1}{2}M'_{\delta_e} & \frac{1}{2}M'_{\delta_f} & \frac{1}{2}M'_{\delta_f} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2}Y'_{\delta_d t} & \frac{1}{2}Y'_{\delta_d t} & -\frac{1}{2}Y'_{\delta_a} & \frac{1}{2}Y'_{\delta_a} & Y'_{\delta_r} \\ -\frac{1}{2}L'_{\delta_d t} & \frac{1}{2}L'_{\delta_d t} & -\frac{1}{2}L'_{\delta_a} & \frac{1}{2}L'_{\delta_a} & L'_{\delta_r} \\ -\frac{1}{2}N'_{\delta_d t} & \frac{1}{2}N'_{\delta_d t} & -\frac{1}{2}N'_{\delta_a} & \frac{1}{2}N'_{\delta_a} & N'_{\delta_r} \end{bmatrix} \tag{3.10}$$

3.4.3 Actuator Design Model

The actuator *design* model is a reduced order model compared to the truth model. Each actuator truth model is a fourth order model, expressed as

$$\frac{\delta_{act}(s)}{\delta_{cmd}(s)} = \frac{(20.2)(144.8)(71.4)^2}{(s+20.2)(s+144.8)(s^2+2(0.736)(71.4)s+71.4^2)} \tag{3.11}$$

where δ_{cmd} is the *commanded* actuator position and δ_{act} is the *actual* actuator position. In order to simplify the model, a first-order model is chosen[14]. Classical control theory might place the break point at $\omega = 20.2\ \frac{rad}{sec}$; however, Eide suggested that placing the break point at $\omega = 14\ \frac{rad}{sec}$ was a better fit to the truth model within the bandwidth of interest [10,11]. To be consistent with results, this research will also use a first order lag with a break point at $\omega = 14\ \frac{rad}{sec}$, expressed as

$$\frac{\delta_{act}(s)}{\delta_{cmd}(s)} = \frac{14}{s+14} \tag{3.12}$$

[14]Research has shown zero-order models for actuators drastically degrade performance [35].

The design model of the actuators in state space form is then

$$\dot{\delta}_{act}(t) = -14 \cdot \mathbf{I}_{5x5} \cdot \delta_{act}(t) + 14 \cdot \mathbf{I}_{5x5} \cdot \delta_{cmd}(t) \tag{3.13}$$

where

$$\delta_{act}(t) = \begin{bmatrix} \delta_{ls}(t) \\ \delta_{rs}(t) \\ \delta_{lf}(t) \\ \delta_{rf}(t) \\ \delta_{r}(t) \end{bmatrix} \quad \text{and} \quad \delta_{cmd}(t) = \begin{bmatrix} \delta_{ls_{cmd}}(t) \\ \delta_{rs_{cmd}}(t) \\ \delta_{lf_{cmd}}(t) \\ \delta_{rf_{cmd}}(t) \\ \delta_{r_{cmd}}(t) \end{bmatrix} \tag{3.14}$$

3.4.4 Augmented System

Now that a reduced order actuator model is defined, the actuator states must be augmented to the original aircraft system state vector. Notice that the definition of δ_{act} in Equation (3.14) is the same as that of $\mathbf{u}_{mod}$ in Equation (3.6). Thus, using Equation (3.9) and noting $\delta_{act} = \mathbf{u}_{mod}$, Equation (3.1) can be rewritten as

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}_{mod}\mathbf{u}_{mod}(t) + \mathbf{G}\mathbf{w}(t) \\ &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}_{mod}\delta_{act}(t) + \mathbf{G}\mathbf{w}(t) \end{aligned} \tag{3.15}$$

If new augmented state and input vectors are defined as

$$\mathbf{x}_{aug}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \delta_{act}(t) \end{bmatrix}, \quad \mathbf{u}_{aug}(t) = [\delta_{cmd}(t)] \tag{3.16}$$

then Equations (3.13) and (3.15) could be augmented as follows:

$$\begin{array}{rcccccccc} \dot{\mathbf{x}}(\mathbf{t}) & = & \mathbf{A}\mathbf{x}(t) & + & \mathbf{B}_{mod}\delta_{act}(t) & + & \mathbf{0} \cdot \delta_{cmd}(t) & + & \mathbf{G}\mathbf{w}(t) \\ \dot{\delta}_{act}(t) & = & \mathbf{0}\mathbf{x}(t) & - & 14 \cdot \mathbf{I}_{5x5} \cdot \delta_{act}(t) & + & 14 \cdot \mathbf{I}_{5x5} \cdot \delta_{cmd}(t) & + & \mathbf{0}\mathbf{w}(t) \end{array}$$

$$\dot{\mathbf{x}}_{aug}(t) = \begin{bmatrix} \mathbf{A} & \mathbf{B}_{mod} \\ \mathbf{0} & -14 \cdot \mathbf{I} \end{bmatrix} \mathbf{x}_{aug}(t) + \begin{bmatrix} \mathbf{0} \\ 14 \cdot \mathbf{I} \end{bmatrix} \mathbf{u}_{aug}(t) + \begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix} \mathbf{w}(t)$$

The continuous-time system model is now complete. It is of the form

$$\dot{\mathbf{x}}_{aug}(t) = \mathbf{A}_{aug}\mathbf{x}_{aug}(t) + \mathbf{B}_{aug}\mathbf{u}_{aug}(t) + \mathbf{G}_{aug}\mathbf{w}(t) \tag{3.17}$$

where the augmented state and input vectors are defined in Equation (3.16) and the augmented plant, input, and noise injection matrices are defined as

$$\mathbf{A}_{aug} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_{mod} \\ \mathbf{0} & -14 \cdot \mathbf{I} \end{bmatrix}, \quad \mathbf{B}_{aug} = \begin{bmatrix} \mathbf{0} \\ 14 \cdot \mathbf{I} \end{bmatrix}, \quad \mathbf{G}_{aug} = \begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix} \tag{3.18}$$

Having the complete, continuous-time design model, Section 3.4.5 now seeks an equivalent, discrete-time model.

3.4.5 *Equivalent, Discrete-Time Model*

Because this research is implemented on a digital flight computer, the model in Equation (3.17) cannot be used directly. Either the results from using a continuous-time filter based on the continuous-time model of Equation (3.17) must be discretized or an equivalent, discrete-time model must be used to synthesize a sample–data filter. Maybeck [22] has shown that the first approach yields only an approximation to the optimal, discrete-time filter; moreover, the first method is more difficult to implement. Thus, an equivalent discrete-time model is sought. The basic concept of such a model is that it should generate a discrete-time state vector output that is indistinguishable from the result of putting the continuous-time model's state through a sampler operating at a specified sampling rate.

The complete development of the following equations is given in Maybeck [22]. An equivalent, discrete-time model is of the form

$$\mathbf{x}_{aug}(t_{i+1}) = \Phi \mathbf{x}_{aug}(t_i) + \mathbf{B}_{d-aug}\mathbf{u}_{aug}(t_i) + \mathbf{w}_{d-aug}(t_i) \tag{3.19}$$

such that

$$\Phi = e^{\mathbf{A}_{aug}\Delta T} \tag{3.20}$$

$$\mathbf{B}_{d-aug} = \left(\int_0^{\Delta T} e^{\mathbf{A}_{aug}\tau} d\tau\right) \mathbf{B}_{aug} \tag{3.21}$$

where ΔT is the sample period. The appropriate discrete-time white noise, $\mathbf{w}_{d-aug}$, has the statistics

$$\begin{aligned} E\{\mathbf{w}_d(t_i)\} &= \mathbf{0} \\ E\left\{\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_i)\right\} &= \mathbf{Q}_{d-aug} = \int_0^{\Delta T} e^{\mathbf{A}_{aug}\tau}\mathbf{G}_{aug}\mathbf{Q}\mathbf{G}_{aug}^T e^{\mathbf{A}_{aug}^T\tau} d\tau \\ E\left\{\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_j)\right\} &= \mathbf{0}, \quad t_i \neq t_j \end{aligned} \tag{3.22}$$

3.4.6 *Discrete-Time Measurement Model*

The VISTA F-16 digital flight computer receives data-sampled, noise-corrupted measurements at a sample rate of 64 Hz. The measurements can be expressed as

$$\mathbf{z}(t_i) = \mathbf{H}_{aug}\mathbf{x}_{aug}(t_i) + \mathbf{v}(t_i) \tag{3.23}$$

where $\mathbf{z}$ is the measurement vector, $\mathbf{H}_{aug}$ is measurement matrix, $\mathbf{x}_{aug}$ is the augmented state vector, and $\mathbf{v}$ is measurement noise.

3.4.6.1 Measurement Vector.

Table 6. Measurement Vector Description

State Vector
$\mathbf{z}(t_i) = \begin{bmatrix} \alpha(t_i) \\ q(t_i) \\ a_n(t_i) \\ p(t_i) \\ r(t_i) \\ a_y(t_i) \end{bmatrix}$

Symbol	Description	Units
α	Angle of Attack	rad
q	Pitch Rate	rad/sec
a_n	Normal Acceleration*	g's
p	Roll Rate	rad/sec
r	Yaw Rate	rad/sec
a_y	Lateral Acceleration*	g's

*measured at the pilot's station

The measurement vector, $\mathbf{z}(t_i)$, defined in Equation (3.23) is described in Table 6. All quantities except for a_n and a_y are available as states from the system model. Eide [10,11] and Stepaniak [41,42] found linear models for these accelerations (expressed at the pilot's station):

$$a_n = -\frac{u}{g}(\dot{\alpha} - q) + \frac{l_x}{g}\dot{q} \tag{3.24}$$

$$a_y = -\frac{u}{g}\left(\dot{\beta} + r\right) - \phi + \frac{l_x}{g}\dot{r} \tag{3.25}$$

where gravity is $g = 32.17\frac{ft}{sec^2}$ and l_x is the distance from the aircraft's center of gravity to the pilot's station calculated as

$$l_x = 9.988 + 0.01 \cdot \bar{c} \cdot x_{cg} \tag{3.26}$$

where the mean aerodynamic chord is $\bar{c} = 11.32\ ft$ and the *average* location of the center of gravity from the origin of the body axes in the x-axis direction (all other directions are assumed negligible) is $x_{cg} = 37.376\ ft$. The term ϕ has units of g's; it is derived from the small angle approximation of $g \cdot \sin(\phi)$, where the gravity term is divided out to form the units of g's. When the linear model for a_n as expressed in Equation (3.24) was compared to the non-linear truth model, it provided very satisfactory results [10, 11, 41, 42]. However, the linear model for a_y expressed in Equation (3.25) compared to the non-linear truth model did not provide satisfactory results [10, 11, 41, 42]. After further analysis by Eide and Stepaniak, the non-linear *truth* model was replaced with a linear model. For comparison and time considerations, this research effort also incorporates the linear model given by Equation (3.25) as the *truth* model for lateral acceleration at the pilot's station.

3.4.6.2 Measurement Matrix. Recalling that $\mathbf{x}_{aug} = \left[\begin{array}{cc} \mathbf{x}^T & \delta_{act}^T \end{array} \right]^T$, the measurement matrix, $\mathbf{H}_{aug}$ of Equation (3.23), can be conveniently expressed as

$$\mathbf{H}_{aug} = \left[\begin{array}{cc} \mathbf{H} & \mathbf{D}_z \end{array} \right] \tag{3.27}$$

so that

$$\mathbf{H}_{aug}\mathbf{x}_{aug} = \left[\begin{array}{cc} \mathbf{H} & \mathbf{D}_z \end{array} \right] \left[\begin{array}{c} \mathbf{x} \\ \delta_{act} \end{array} \right] = \mathbf{H}\mathbf{x} + \mathbf{D}_z\delta_{act} \tag{3.28}$$

Thus, Equations (3.24) and (3.25) can be expressed in the form of Equation (3.28) by defining

$$\mathbf{H} = \left[\begin{array}{cccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{\bar{U}}{32.2}Z'_\theta & -\frac{\bar{U}}{32.2}Z'_u & -\frac{\bar{U}}{32.2}Z'_\alpha & -\frac{\bar{U}}{32.2}\left(Z'_q - 1\right) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{\bar{U}}{32.2}Y'_\phi - 1 & \frac{\bar{U}}{32.2}Y'_\beta & \frac{\bar{U}}{32.2}Y'_p & \frac{\bar{U}}{32.2}\left(Y'_r - 1\right) \end{array} \right] \tag{3.29}$$

$$\mathbf{D}_z = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{\bar{U}}{32.2}\left(\frac{1}{2}Z'_{\delta_e}\right) & -\frac{\bar{U}}{32.2}\left(\frac{1}{2}Z'_{\delta_e}\right) & -\frac{\bar{U}}{32.2}\left(\frac{1}{2}Z'_{\delta_f}\right) & -\frac{\bar{U}}{32.2}\left(\frac{1}{2}Z'_{\delta_f}\right) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\bar{U}}{32.2}\left(-\frac{1}{2}Y'_{\delta_{dt}}\right) & \frac{\bar{U}}{32.2}\left(\frac{1}{2}Y'_{\delta_{dt}}\right) & \frac{\bar{U}}{32.2}\left(-\frac{1}{2}Y'_{\delta_a}\right) & \frac{\bar{U}}{32.2}\left(\frac{1}{2}Y'_{\delta_a}\right) & \frac{\bar{U}}{32.2}\left(Y'_{\delta_r}\right) \end{array} \right] \tag{3.30}$$

where $\bar{U}$ is the velocity vector at trim.

3.4.6.3 Measurement Noise Vector.

Table 7. Measurement Noise Covariance Description

R	RMS Noise		Units
R(1,1)	α	0.004	rad
R(2,2)	q	0.006	rad/sec
R(3,3)	a_n	0.01	g's
R(4,4)	p	0.02	rad/sec
R(5,5)	r	0.006	rad/sec
R(6,6)	a_y	0.005	g's

The measurement noise vector, $\mathbf{v}\left(t_i\right)$, as defined in Equation (3.23) is considered discrete white Gaussian noise, with a mean of zero and covariance $\mathbf{R}$ as given in Table 7. From the definition of $\mathbf{R}$, it can be seen that $\mathbf{v}\left(t_i\right)$ is a $6x1$ vector and that all noises are assumed uncorrelated. Note that the table presents $\mathbf{R}\left(j,j\right)^{\frac{1}{2}}$ values for $j = 1, 2, \cdots, 6$, rather than $\mathbf{R}\left(j,j\right)$ values.

3.4.7 Summary

At this point, the discrete-time system and measurement design model is complete. From Equations (3.19) and (3.23), the design model is expressed as

$$\mathbf{x}_{aug}(t_{i+1}) = \Phi \mathbf{x}_{aug}(t_i) + \mathbf{B}_{d-aug}\mathbf{u}_{aug}(t_i) + \mathbf{w}_{d-aug}(t_i) \tag{3.31}$$

$$\mathbf{z}(t_i) = \mathbf{H}_{aug}\mathbf{x}_{aug}(t_i) + \mathbf{v}(t_i) \tag{3.32}$$

The augmented state and input vectors, $\mathbf{x}_{aug}$ and $\mathbf{u}_{aug}$, are defined in Equation (3.16). The state transition matrix, Φ, is found through Equation (3.20) using $\mathbf{A}_{aug}$ given in Equation (3.18). The discrete-time input matrix, $\mathbf{B}_{d-aug}$, is found through Equation (3.21) using $\mathbf{A}_{aug}$ and $\mathbf{B}_{aug}$ given in Equation (3.18). The discrete-time dynamics driving noise, $\mathbf{w}_{d-aug}(t_i)$, is white, Gaussian, zero-mean, and with strength $\mathbf{Q}_{d-aug}$ found through Equation (3.22) using $\mathbf{A}_{aug}$ and $\mathbf{G}_{aug}$ given in Equation (3.18) and the continuous-time strength $\mathbf{Q}$ given in Table 4. The measurement vector, $\mathbf{z}$, is defined in Table 6. The measurement matrix, $\mathbf{H}_{aug}$, is defined in Equation (3.27), where $\mathbf{H}$ is given in Equation (3.29) and $\mathbf{D}_z$ is given in Equation (3.30). Finally, the discrete-time measurement noise, $\mathbf{v}$, is white, Gaussian, zero-mean, and with covariance $\mathbf{R}$ given in Table 7.

3.5 Failure Models

In order to apply the MMAE architecture to failure detection and estimation, failures must be incorporated into the truth model as well as the design model. It is convenient to express failures in terms of an *effectiveness factor*, ϵ, with a range

$$0 \leq \epsilon \leq 1 \tag{3.33}$$

An effectiveness of 0% ($\epsilon = 0$) indicates a 100% failure, or a "full failure". Conversely, an effectiveness of 100%, ($\epsilon = 1$), indicates a 0% failure, or "no failure". Effectiveness values $0 < \epsilon < 1$ indicate "partial failures".

3.5.1 Truth Model Failures

The truth model must be modified to incorporate failures. For the truth model, it is desirable to model the failure as it would occur in the "real world" on the VISTA F-16. Due to the lack of a complete failure modes study, the research done in this area can only speculate what might occur.

For sensor failures, the nonlinear truth model can model failures in various ways. Sensor failures can be modeled as the loss of the desired noise-free signal portion of the sensor reading while still admitting sensor noise into the measurement. This is the approach taken for the truth model used in this research. In reality, other sensor failures that may occur include sensor bias, increased sensor noise levels, or changes in the $\mathbf{H}$ matrix.

For actuator failures, modeling is more difficult. Actuator failures are modeled by multiplying the *commanded* actuator position by ϵ. For full failures ($\epsilon = 0$), the appropriate surface would be commanded to its trim position. This boundary condition is desirable because it approximates "failure to free stream". For no failures ($\epsilon = 1$), the appropriate surface would be given the full command, ensuring consistency during normal flight conditions. For partial failures ($0 < \epsilon < 1$), scaling the command by ϵ results in the effect of reduced command authority. These three conditions are modeled in the truth model used in this research because they represent the characteristics of a failed actuator without the need for determining the effect of a failed surface on the aerodynamic characteristics of the aircraft. If time permitted, actuator failures could be more accurately modeled by additionally applying changes in the stability derivatives, effecting the $\mathbf{A}$ matrix in addition to the $\mathbf{B}$ and $\mathbf{H}$ matrices. For instance, battle damage to one actuator may emulate stability derivatives representing an asymmetric aircraft. Other failure characteristics not modeled may also be experienced, such as stuck actuators (rather than failure to free stream), removal of portions of an actuator (as opposed to reduced effectiveness of a given surface), or a deterioration in the actuator's time response.

3.5.2 Design Model Failures

In the linear design model, Stepaniak [41,42] introduced the use of failure matrices $\mathbf{F}_{ai}$ and $\mathbf{F}_{sj}$, where the subscripts a and s indicate an actuator or sensor failure, respectively, and i and j indicate the index of the actuator or sensor failure, respectively. The failure matrices $\mathbf{F}_{ai}$ and $\mathbf{F}_{sj}$ are identity matrices except that $\mathbf{F}_{ai}(i,i) = \epsilon$ if the i^{th} actuator has failed and $\mathbf{F}_{sj}(j,j) = \epsilon$ if the j^{th} sensor has failed.

For actuator failures, the failure matrix, $\mathbf{F}_{ai}$, is inserted into the *continuous-time* system model *after* augmenting the actuator states. This is done to reflect the fact that the truth model uses continuous-time dynamics. The design model can be expressed as

$$\dot{\mathbf{x}}_{aug}(t) = \mathbf{A}_{aug}\mathbf{x}_{aug}(t_i) + \mathbf{B}_{aug}\mathbf{F}_{ai}\mathbf{u}_{aug}(t) + \mathbf{G}_{aug}\mathbf{w}(t) \tag{3.34}$$

From Equation (3.34) it can be seen that a single actuator failure affects a single *column* of the $\mathbf{B}_{aug}$ matrix. Note that the *commanded* actuator position is multiplied by ϵ and that full failures ($\epsilon = 0$) are treated as a failure to "free stream". Further, to obtain an implementable failure design model, an equivalent, discrete-time model must be found. The only change is in $\mathbf{B}_{d-aug}$, where now instead of Equation (3.21), $\mathbf{B}_{d-aug}$ is expressed as

$$\mathbf{B}_{d-aug} = \left(\int_0^{\Delta T} e^{\mathbf{A}_{aug}\tau} d\tau \right) \mathbf{B}_{aug}\mathbf{F}_{ai} \tag{3.35}$$

For sensor failures, the failure matrix, $\mathbf{F}_{sj}$, is inserted into the sampled-data measurement model as

$$\mathbf{z}(t_i) = \mathbf{F}_{sj}\mathbf{H}_{aug}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{3.36}$$

Equation (3.36) displays the fact that a single sensor failure affects a single *row* of the $\mathbf{H}_{aug}$ matrix. Thus, sensor failures are modelled as the loss of information while still admitting sensor noise into the measurement vector. Note that partial sensor failures are *not* modelled in this fashion; that is, the effectiveness factor may only equal zero for full sensor failures or one for no sensor failure. Although this research effort does not consider partial sensor failures, an appropriate partial sensor failure model may be characterized by an increase in the sensor noise.

3.6 Control Redistribution

This section reviews Stepaniak's [41, 42] development of Control Redistribution (CR). CR was developed by Stepaniak to be used in conjunction with MMAE-based control when it became apparent that Multiple Model Adaptive Control (MMAC) could not perform sufficiently due to numerical problems with the LQ synthesis of multiple elemental controllers required within the MMAC. The remainder of this section presents CR in Section 3.6.1, details its application to full failures and no failures in Section 3.6.2, and presents a new *modified* CR scheme used to handle

partial failures in Section 3.6.3. The interested reader is referred to Clark's thesis [8] for a detailed history of other control redistribution efforts.

3.6.1 CR Algorithm Development

Control Redistribution (CR) uses the *inherent redundancy* existing in an aircraft's suite of control surfaces to restore control authority if an impairment to a control surface occurs. CR is ideally suited for fly-by-wire aircraft in which a flight computer determines what control vector command to send to the actuators. For instance, the VISTA F-16 receives longitudinal input from the pilot in terms of C^*, a scheduled blending of pitch rate and normal acceleration based on dynamic pressure [36]. Based on the commanded C^* and dynamic pressure, the flight computer determines what control authority to issue to the actuators. If an actuator is impaired, CR allows the flight computer to redistribute the control to the other unimpaired actuators to achieve the desired aircraft behavior, rather than relying on the pilot to compensate for the actuator impairment. If an algorithm has detected a failure and estimated its extent, the pilot can issue C^*commands as usual and CR will compensate for the surface impairment in a manner that is transparent to the pilot. Of course, implementation could provide cockpit displays that inform the pilot of failures.

Mathematically, the aircraft performance before and after a control surface impairment should be equal:

$$\begin{aligned} \dot{\mathbf{x}}_{before} &= \dot{\mathbf{x}}_{after} \\ \mathbf{Ax}+\mathbf{Bu} &= \mathbf{A}_{fail}\mathbf{x}+\mathbf{B}_{fail}\mathbf{u}_r \end{aligned} \tag{3.37}$$

where $\mathbf{A}_{fail}$ and $\mathbf{B}_{fail}$ are the plant and input matrices after the failure, respectively, and $\mathbf{u}_r$ is the *redistributed* control vector. As described in the failure truth model development in Section 3.5.1, the plant matrix before a failure and the plant matrix after a failure are assumed to be approximately equal:

$$\mathbf{A}_{fail} \approx \mathbf{A} \tag{3.38}$$

Thus, from Equation (3.37),

$$\mathbf{B}_{fail}\mathbf{u}_r \approx \mathbf{Bu} \tag{3.39}$$

must be satisfied. From Section 3.5.2, the input matrix after a failure is equal to the input matrix before a failure post-multiplied by the failure matrix, $\mathbf{F}_{ai}$, expressed as

$$\mathbf{B}_{fail} = \mathbf{B}\mathbf{F}_{ai} \tag{3.40}$$

The redistributed control vector can be defined in terms of the original control vector as

$$\mathbf{u}_r \triangleq \mathbf{D}_{ai}\mathbf{u} \tag{3.41}$$

by introducing the control redistribution matrix, $\mathbf{D}_{ai}$, where the subscript a denotes actuator and the subscript i indicates the i^{th} actuator. The control redistribution matrix must be computed on-line or known *a-priori*. By substitution and manipulation, an expression for $\mathbf{D}_{ai}$ is obtained[15]:

$$\mathbf{B}_{fail}\mathbf{u}_r = \mathbf{B}\mathbf{u} \tag{3.42}$$

$$(\mathbf{B}\mathbf{F}_{ai})(\mathbf{D}_{ai}\mathbf{u}) = \mathbf{B}\mathbf{u} \tag{3.43}$$

$$\mathbf{B}\mathbf{F}_{ai}\mathbf{D}_{ai} = \mathbf{B} \tag{3.44}$$

$$\mathbf{D}_{ai} = (\mathbf{B}\mathbf{F}_{ai})^{+}\mathbf{B} \tag{3.45}$$

where $\mathbf{u}$ is known and eliminated from both sides of Equation (3.43) and the superscript $+$ in Equation (3.45) indicates the Moore-Penrose [44] pseudoinverse. The Moore-Penrose pseudoinverse yields a true inverse if $\mathbf{B}\mathbf{F}_{ai}$ is full rank, shown as

$$\mathbf{D}_{ai} = (\mathbf{B}\mathbf{F}_{ai})^{+}\mathbf{B} = (\mathbf{B}\mathbf{F}_{ai})^{-1}\mathbf{B} = \mathbf{F}_{ai}^{-1}$$

In the case where $\mathbf{B}\mathbf{F}_{ai}$ is rank deficient, the Moore-Penrose pseudoinverse yields the best solution in the least squares sense, that is, the error $\|\mathbf{B}\mathbf{F}_{ai}\mathbf{D}_{ai} - \mathbf{B}\|$ is minimized [44].

3.6.2 CR Application

To apply control redistribution to the VISTA F-16, the control redistribution matrix, $\mathbf{D}_{ai}$, given in Equation (3.45) must be explored, considering the control vector actuators given in Equation (3.14) and (3.16) defined in Table ?? and given the following indices:

$$\mathbf{u}_{aug} = \begin{bmatrix} \delta_{ls_{cmd}} \\ \delta_{rs_{cmd}} \\ \delta_{lf_{cmd}} \\ \delta_{rf_{cmd}} \\ \delta_{r_{cmd}} \end{bmatrix}, \quad i = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \tag{3.46}$$

[15] The approximation symbol $\approx$ indicating that $\mathbf{A}_{fail} \approx \mathbf{A}$ is dropped.

For the fully functional case (and knowing that there are five actuators for the VISTA F-16), the failure matrix is $\mathbf{F}_{ai} = \mathbf{I}_{5x5}$, and from Equation (3.45), $\mathbf{D}_{ai}^{\epsilon=1} = \mathbf{I}_{5x5}$, where the superscript ϵ is placed on $\mathbf{D}_{ai}$ to indicate its corresponding effectiveness value. This boundary condition is intuitive; CR simply passes $\mathbf{u}$ as $\mathbf{u}_r$ when no failure has occurred. For the fully failed case, $\mathbf{D}_{ai}$ is known *a-priori* (for the VISTA F-16). Stepaniak [41,42] lists $\mathbf{D}_{ai}$ for a fully failed left stabilator as

$$\mathbf{D}_{a1}^{\epsilon=0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0.9060 & 0 & 1 & 0 & 0 \\ -0.9060 & 0 & 0 & 1 & 0 \\ -0.7862 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

(Recall from Equation (3.41) that $\mathbf{u}_r \triangleq \mathbf{D}_{ai}\mathbf{u}$, where $\mathbf{u}$ is the original control vector and $\mathbf{u}_r$ is the redistributed control vector.) Looking at the first row of $\mathbf{D}_{a1}^{\epsilon=0}$, no control is placed on the failed left stabilator. Looking at the remaining rows of $\mathbf{D}_{a1}^{\epsilon=0}$, control for the unfailed actuators is a combination of original control. Therefore, $\mathbf{D}_{ai}$ *redistributes* the control authority from the failed actuator to the unfailed actuators to produce an *equivalent* control vector. Although *equivalence* was proven mathematically, problems can arise when trying to implement this scheme for partial failures (see Section 3.6.3).

Notice the form of $\mathbf{D}_{a1}^{\epsilon=0}$ in Equation (3.47). The only information in the $\mathbf{D}_{ai}$ matrix is the i^{th} column. Thus, a compact notation can be used to incorporate all five $\mathbf{D}_{ai}$ matrices for full failures into one matrix. Stepaniak [41,42] found for the VISTA F-16,

$$\mathbf{D}_{a}^{\epsilon=0} = \begin{bmatrix} 0 & 1 & 1.1037 & -1.1037 & -1.2719 \\ 1 & 0 & -1.1037 & 1.1037 & 1.2719 \\ 0.9060 & -0.9060 & 0 & 1 & 1.1524 \\ -0.9060 & 0.9060 & 1 & 0 & -1.1524 \\ -0.7862 & 0.7862 & 0.8678 & -0.8678 & 0 \end{bmatrix} \tag{3.48}$$

Now, any of the five $\mathbf{D}_{ai}^{\epsilon=0}$ matrices can be formed by starting with the identity matrix and replacing the i^{th} column with the i^{th} column of the $\mathbf{D}_{a}^{\epsilon=0}$ matrix given in Equation (3.48). This compact notation is useful because it reduces data storage requirements during implementation. It is important to notice that the zeros along the diagonal cause no control authority to be issued to the failed

actuator, while the off-diagonal terms cause the control authority to be *distributed* to the unfailed actuators.

3.6.3 Modified CR

When trying to *implement* the $\mathbf{D}_{ai}$ matrix for partial failures, problems can arise. In the partial failure case, the failure matrix $\mathbf{F}_{ai}$ is full rank. Therefore, the true inverse exists and $\mathbf{D}_{ai}$ can be expressed as

$$\mathbf{D}_{ai}^{\epsilon=(0,1]} = \mathbf{F}_{ai}^{-1} \tag{3.49}$$

Notice this holds for ϵ contained within the range $(0,1]$. For $\epsilon = 1$, $\mathbf{I}^{-1} = \mathbf{I}$ and no problems arise. However, for ϵ in the range $(0,1)$, $\mathbf{F}_{ai}$ is the identity matrix, except the i^{th} diagonal entry is equal to ϵ. Thus, from Equation (3.49), $\mathbf{D}_{ai}^{\epsilon=(0,1)}$ is the identity matrix, except the i^{th} diagonal entry is equal to ϵ^{-1}. This makes sense intuitively: "If the i^{th} actuator is only ϵ effective, increase its gain by ϵ^{-1}." However, this result is unsatisfactory because it results in actuator saturation and rate limits [8,41,42] as well as strong commands to a partially failed actuator.

To avoid this situation, a *modified* Control Redistribution technique is used. In the case of partial failures, Equation (3.45) is *not* used; rather, the following approach is taken. The modified redistributed control vector for partial failures is defined as

$$\mathbf{u}_{r_{mod}} = \mathbf{u}|_{\epsilon=1} + (1-\epsilon_0)\,\mathbf{u}_r|_{\epsilon=0} \tag{3.50}$$

where ϵ_0 is the effectiveness value of the actuator undergoing partial failure such that ϵ_0 lies in the range $(0,1)$. The term $\mathbf{u}|_{\epsilon=1}$ is used to indicate that the normal control vector $\mathbf{u}$ is applied, and it is anticipated that the partially failed actuator will exhibit a response as if multiplied by $\epsilon = \epsilon_0$. The first term in Equation (3.50) applies the full original command to the partially failed actuator; the result is that the actuator actually produces an output as though $\mathbf{F}_{ai}^{\epsilon=\epsilon_0}\mathbf{u}$ has been received, where the superscript $\epsilon = \epsilon_0$ on $\mathbf{F}_{ai}$ indicates a failure matrix with an effectiveness factor of $\epsilon = \epsilon_0$. The remaining term in Equation (3.50), $(1-\epsilon_0)\,\mathbf{u}_r|_{\epsilon=0}$ compensates for the lost effectiveness in the failed actuator by redistributing the control authority lost $(1-\epsilon_0)$ to the other actuators through CR; that is, $\mathbf{u}_r|_{\epsilon=0} = \mathbf{D}_{ai}^{\epsilon=0}\mathbf{u}$ as defined in Equation (3.41).

To find an equivalent modified redistribution matrix, $\mathbf{D}_{ai_{mod}}$, the control $\mathbf{u}_{r_{mod}}$ given by Equation (3.50) can be re-written as

$$\mathbf{u}_{r_{mod}} = \mathbf{u}|_{\epsilon=1} + (1-\epsilon_0)\,\mathbf{u}_r|_{\epsilon=0} \tag{3.51}$$

$$= \mathbf{u} + (1-\epsilon)\,\mathbf{D}_{ai}^{\epsilon=0}\mathbf{u} \tag{3.52}$$

$$= \left(\mathbf{I} + (1-\epsilon)\,\mathbf{D}_{ai}^{\epsilon=0}\right)\mathbf{u} \tag{3.53}$$

$$= \mathbf{D}_{ai_{mod}}\mathbf{u} \tag{3.54}$$

Noticing Equation (3.53) and recalling that $\mathbf{D}_{ai}^{\epsilon=0}$ is the identity matrix with the i^{th} column replaced with i^{th} column of $\mathbf{D}_{a}^{\epsilon=0}$ given in Equation (3.48), an expression for $\mathbf{D}_{ai_{mod}}$ given in Equation (3.54) can be written. Expressed in compact notation defining $\gamma = 1-\epsilon$,

$$\mathbf{D}_{a_{mod}}(\gamma) = \begin{bmatrix} 1 & \gamma & 1.1037\gamma & -1.1037\gamma & -1.2719\gamma \\ \gamma & 1 & -1.1037\gamma & 1.1037\gamma & 1.2719\gamma \\ 0.9060\gamma & -0.9060\gamma & 1 & \gamma & 1.1524\gamma \\ -0.9060\gamma & 0.9060\gamma & \gamma & 1 & -1.1524\gamma \\ -0.7862\gamma & 0.7862\gamma & 0.8678\gamma & -0.8678\gamma & 1 \end{bmatrix} \tag{3.55}$$

where now $\mathbf{D}_{ai_{mod}}$ can be formed by taking $\mathbf{I}_{5x5}$ and replacing the i^{th} column with the i^{th} column given by $\mathbf{D}_{a_{mod}}(\gamma)$ in Equation (3.55).

Using $\mathbf{D}_{a_{mod}}(\gamma)$ does increase the load on the other actuators; however, this is reasonable because the other actuators must compensate for the loss of control authority from the failed actuator. The advantage is $\mathbf{D}_{a_{mod}}$ "works" the *unfailed* actuators harder, while $\mathbf{F}_{ai}^{-1}$ "works" the *failed* actuator harder. For example, consider a left stabilator failure with an effectiveness of 10% ($\epsilon = 0.1$). The resultant $\mathbf{u}_r$ is

$$\mathbf{u}_r = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{u}$$

whereas the resultant $\mathbf{u}_{r_{mod}}$ is

$$\mathbf{u}_{r_{mod}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.9 & 1 & 0 & 0 & 0 \\ 0.9060\,(0.9) = 0.8154 & 0 & 1 & 0 & 0 \\ -0.9060\,(0.9) = -0.8154 & 0 & 0 & 1 & 0 \\ -0.7862\,(0.9) = -0.7076 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{u}$$

Notice using CR simply increases the gain by 10 for the failed actuator. Using *modified* CR, however, leaves the gain the same for the failed actuator, but increases the load on the unfailed actuators. Distributing the load *from* the failed actuator *to* the unfailed actuators is an advantage because working a partially failed actuator harder may cause a further decrease in its effectiveness.

When using the modified Control Redistribution approach, caution must be taken in two areas. First, the entire development assumes that if the control $\mathbf{u}$ is applied, then the result will be as if multiplied by ϵ for the failed actuator. This requires that the estimate[16] of ϵ is accurate so that the remaining control added to compensate for the loss is appropriate. If the estimate of ϵ is too high, the failed actuator will offer less control authority than expected and not enough compensating control will be added, causing the aircraft to perform sluggishly. If the estimate of ϵ is too low, the failed actuator will offer more control authority than expected and too much compensating control will be added, causing the aircraft to over-compensate and perhaps causing instabilities. This leads into the second area of caution, use of logic to enhance performance of the modified CR algorithm. Logic should be incorporated that makes decisions based on the value of the estimated ϵ. For example, consider the condition in which the estimate is $\epsilon = 0$. Using *modified* CR, the original control will be issued to the failed actuator anticipating that the response of the failed actuator will be zero. Logic should incorporate the use of CR rather than modified CR in this case, so that control authority is *not* issued to a completely failed actuator. Depending on the application, other regions of effectiveness may also be of concern. For this research effort, an estimate is provided to use in a CR or modified CR technique. A systematic approach to developing the logic used in this research will be explained in Section 5.6.

3.7 Conclusion

This chapter covered the models pertaining to this research, starting with the "real world" VISTA F-16 in Section 3.2. A full six-degree-of-freedom truth model based on the "real world" VISTA F-16 needed to validate the design was described in Section 3.3. A reduced order design

[16]For this research, the estimate will come from an MMAE algorithm.

model needed to build Kalman filters within the MMAE was developed in Section 3.4. Failures were incorporated into the truth model as well as the design model in Section 3.5. Finally, Control Redistribution and modified Control Redistribution (needed for partial failures) were described in Section 3.6. Having the necessary models, the MMAE architecture will be described in the following chapter.

Chapter 4 - Filter Spawning

4.1 Chapter Overview

This chapter introduces filter spawning as a technique to enhance the application of MMAE to failure detection and estimation. In Section 4.2, the filter models for each Kalman filter in the MMAE algorithm will be reviewed. In Section 4.3, filter spawning is motivated and its basic operation is outlined. In Section 4.4, filter spawning implementation is discussed as it is applied to this research. The chapter closes with a brief summary in Section 4.5.

4.2 Filter Models

In order to apply the Multiple Model Adaptive Estimation (MMAE) algorithm, Kalman filters must be built with models based on various parameter values. For this research, MMAE is applied to the detection and estimation of failures of sensors and actuators. Therefore, filters must be built, each hypothesizing a different failure condition. Recall in Section 3.4 a "design model" was constructed and in Section 3.5 failures were incorporated into the design model via the failure matrices, $\mathbf{F}_{ai}$ and $\mathbf{F}_{sj}$, where the subscripts a and s denote actuator and sensor failures, respectively, and the subscripts i and j denote the failure index of the actuator and sensor, respectively. The design model is expressed as

$$\mathbf{x}_{aug}(t_{i+1}) = \Phi \mathbf{x}_{aug}(t_i) + \mathbf{B}_{d-aug}\mathbf{u}_{aug}(t_i) + \mathbf{w}_{d-aug}(t_i) \tag{4.1}$$

$$\mathbf{z}(t_i) = \mathbf{F}_{sj}\mathbf{H}_{aug}\mathbf{x}_{aug}(t_i) + \mathbf{v}(t_i) \tag{4.2}$$

where $\mathbf{F}_{ai}$ is incorporated into the definition of $\mathbf{B}_{d-aug}$, as defined in Equation (3.35).

For this research, the conditions of no failure, complete actuator failures, complete sensor failures, and partial actuator failures are considered. Thus, a model for each case must be formed

Table 8. Actuator Indices

Index, i	Actuator
1	Left Stabilator
2	Right Stabilator
3	Left Flaperon
4	Right Flaperon
5	Rudder

based on Equations (4.1) and (4.2). The *no failure model* is formed using Equation (4.1), where $\mathbf{B}_{d-aug}$ is based on $\mathbf{F}_{ai} = \mathbf{I}_{5x5}$, and using Equation (4.2) where $\mathbf{F}_{sj} = \mathbf{I}_{6x6}$. The *actuator failure models*, with the actuator index assignment given in Table 8, are formed using Equation (4.1), where $\mathbf{B}_{d-aug}$ is based on $\mathbf{F}_{ai} = \mathbf{I}_{5x5}$ except that the i^{th} diagonal term is replaced with zero, and using Equation (4.2) where $\mathbf{F}_{sj} = \mathbf{I}_{6x6}$. The *sensor failure models*, with the index assignment given in Table 9, are formed using Equation (4.1), where $\mathbf{B}_{d-aug}$ is based on $\mathbf{F}_{ai} = \mathbf{I}_{5x5}$, and using Equation (4.2) where $\mathbf{F}_{sj} = \mathbf{I}_{6x6}$ except that the j^{th} diagonal term is replaced with a zero. The *partial actuator failure models* are formed using Equation (4.1), where $\mathbf{B}_{d-aug}$ is based on $\mathbf{F}_{ai} = \mathbf{I}_{5x5}$ but with the i^{th} diagonal term replaced with the effectiveness value ϵ, and using Equation (4.2) where $\mathbf{F}_{sj} = \mathbf{I}_{6x6}$. (Recall that the effectiveness, ϵ, is used to describe the extent of the partial failure; 100ϵ would be the percent effectiveness and $100\,(1-\epsilon)$ would be the percent failure.) In Section 4.4.1, these models will be used to form the MMAE algorithm.

Table 9. Sensor Indices

Index, j	Sensor
1	Angle of Attack Sensor
2	Pitch Rate Sensor
3	Normal Acceleration Sensor
4	Roll Rate Sensor
5	Yaw Rate Sensor
6	Lateral Acceleration Sensor

4.3 Filter Spawning Conceptualization

Filter spawning is the method used in this research to enhance the estimation of the effectiveness value for a partial actuator failure. Section 4.3.1 will motivate its use by explaining the shortcoming of prior efforts. Section 4.3.2 will explain how filter spawning improves estimation.

4.3.1 Filter Spawning Motivation

Prior research has explored the *detection* and *estimation* of partial actuator failures without the use of spawning. One method used was to blend the fully functional filter and the complete failure filter to obtain an estimate of the effectiveness value [8]. Using this method, the failure is

detected using the maximum a posteriori (MAP) approach, expressed as

$$\hat{\mathbf{a}}_{MMAE}^{MAP}(t_i) = \mathbf{a}_j(t_i) \quad \text{where} \quad j = \arg\{k\max[p_k(t_i)]\} \tag{4.3}$$

where $\hat{\mathbf{a}}_{MMAE}^{MAP}$ is the failure estimate, $\mathbf{a}_j$ is the hypothesized failure for the j^{th} filter, and $p_k(t_i)$ is the conditional probability for the k^{th} filter as computed in Equation (2.19). In words, the filter's hypothesis with the highest conditional probability is the detected failure. If the filter with the highest conditional probability corresponds to the filter hypothesizing a fully functional aircraft, then no failure is detected. If the filter with the highest conditional probability corresponds to a filter hypothesizing a sensor failure, then the detected failure is a complete[17] sensor failure. If, however, the filter with the highest conditional probability corresponds to a filter hypothesizing an actuator failure, then the algorithm declares a *detected* actuator failure and attempts to *estimate* the extent of the failure (in terms of effectiveness).

The actuator failure is *estimated* using a Bayesian blending approach using the MMAE-computed hypothesis conditional probabilities, expressed as

$$\hat{\epsilon}(t_i) = \frac{\epsilon_1 \cdot p_1(t_i) + \epsilon_j \cdot p_j(t_i)}{p_1(t_i) + p_j(t_i)} \tag{4.4}$$

where $\hat{\epsilon}$ is the effectiveness estimate, ϵ_1 is the effectiveness hypothesized by the fully functional filter ($\epsilon_1 = 1$), j is index of the filter with the highest conditional probability as defined in Equation (4.3) (which corresponds to an actuator), ϵ_j is the effectiveness hypothesized by the j^{th} filter (and because only complete actuator failures are modelled, $\epsilon_j = 0$), and p_j is the conditional probability of the j^{th} filter. The denominator term corrects the fact that p_1 and p_j do not sum to one (recall that the sum of *all* the conditional probabilities is one). Noting the values of ϵ_1 and ϵ_j, Equation (4.4) can be easily implemented in the form

$$\hat{\epsilon}(t_i) = \frac{p_1(t_i)}{p_1(t_i) + p_j(t_i)} \tag{4.5}$$

While this method is easy to implement, it yielded only marginal results [8] because Equation (4.4) depends on the probabilities flowing in proportion to the effectiveness. For example, if the true effectiveness was 75%, one might anticipate $p_1 = 0.75$ and $p_j = 0.25$; however, this was not the

[17] Recall that for sensor failures, only complete failures are considered.

case. Because the two models' hypothesized effectiveness values were crudely discretized (the space $[0, 1]$ was discretized with points 0 and 1), the probability tended to flow to the *one* filter with the hypothesis closest in the Baram distance sense. An accurate blended estimate of the effectiveness value could not be found consistently, resulting in an erroneous algorithm-indicated effectiveness estimate. When the hierarchical structure of Section 2.3.4 brought in a new bank of filters based upon this erroneous effectiveness estimate, *none* of the filters in the bank had a very good model of the real world, and all subsequent failure detection performance suffered substantially.

4.3.2 Spawning Filters to Obtain an Estimate

To resolve the discretization problem when *estimating* the extent of an actuator failure, filter spawning incorporates additional filters (based on various levels of effectiveness) to the MMAE bank. In the "filter spawning technique", the *detection* of a failure is accomplished in the same fashion as the previous method, using the MAP estimate as defined in Equation (4.3). Again, if the j^{th} filter corresponds to the fully functional aircraft filter[18], then no failure is declared (where j is defined in Equation (4.3)). If the j^{th} filter corresponds to a sensor failure filter, then a complete sensor failure is declared (where all sensor failures are modelled as complete failures). If the j^{th} filter corresponds to an actuator failure filter, then the algorithm declares a *detected* actuator failure.

After a failure is *detected*, filter spawning is used to enhance *estimation*. Additional filters, called spawned filters, are included into the MMAE bank. Thus, spawning provides for finer parameter discretization among hypothesized filters to yield better parameter estimation, but *only* in the channel corresponding to the detected actuator failure, so as not to require an overwhelming number of elemental filters to be running in parallel at any one time. Each spawned filter's model is based on the actuator failure detected with different effectiveness value in the region $\epsilon = (0, 1)$. Having increased the number of discretization points over the space $\epsilon \in [0, 1]$, the filter spawning technique *estimates* the effectiveness using the Bayesian blending approach, expressed as

[18]For convenience, the filter's hypothesis is used as a desciptor for the filter, i.e., the "fully functional aircraft filter" is the filter hypothesizing a fully functional aircraft. Likewise, "sensor failure filter" and "actuator failure filter" will be used.

$$\hat{\epsilon}(t_i) = \frac{\sum_{k \in \mathcal{K}} [\epsilon_k \cdot p_k(t_i)]}{\sum_{k \in \mathcal{K}} p_k(t_i)} \tag{4.6}$$

where $\hat{\epsilon}$ is the effectiveness estimate, ϵ_k is the effectiveness hypothesized by the k^{th} filter, p_k is the conditional probability of the k^{th} filter, and $\mathcal{K}$ contains the indices to the fully functional filter, the complete actuator failure filter (i.e., j), and the spawned filters. The denominator term corrects the fact that sum of these particular p_k's do not sum to one.

The filter spawning technique improves the effectiveness estimate in two ways. First, because there are partial failure hypotheses, filter spawning does not depend completely on blending to provide an estimate of the effectiveness. That is, one could spawn filters based on effectiveness values of interest to obtain an estimate of effectiveness without blending, as by the MAP method used by Equation (4.3) in detection. Second, blending can be applied because the probabilities are more evenly distributed. Probabilities are more evenly distributed because the hypotheses are "closer" together using the finer discretization, i.e., the space $[0, 1]$ is discretized with points at 0, 1, and the ϵ values hypothesized by the spawned filters rather than just 0 and 1.

4.4 Filter Spawning Implementation

This section presents MMAE with Filter Spawning in detail sufficient for the reader to duplicate its use (as it is used in this research). MMAE with Filter Spawning is accomplished borrowing the hierarchical structure, described in Section 2.3.4. It is important to note this is not necessarily the most *efficient* implementation; rather, the hierarchical structure was chosen for time considerations because it is *easier* to implement than a more generic moving-bank structure [5,14,16,19,20,25,47]. Furthermore, while filter spawning is ideally suited to MMAE algorithms that can expand and contract the bank of parameter values upon which elemental filters are based, efficient implementation involves dynamic memory allocation. However, application using dynamic memory allocation is left for future consideration, whether or not the hierarchical structure is employed.

The flow chart for the MMAE with Filter Spawning algorithm used in this research is shown in Figure 6. In Section 4.4.1, the initialization block is discussed. Following initialization, the "Conventional MMAE Computation" group performs computation standard to all MMAE algorithms, as discussed in Section 4.4.2. To apply MMAE to the task of this research, logic that performs "Failure Detection, Estimation, and Control" is discussed in Section 4.4.3. To complete the flow chart, the "Bank Swapping" group is discussed in Section 4.4.4.

4.4.1 Initialization

The "Initialize" block involves two steps. The models and banks used by the MMAE algorithm must be placed into memory, and initial conditions must be specified.

4.4.1.1 Bank Definition. The algorithm uses one bank consisting of 12 filters as well as five additional banks consisting of 15 filters each, as described in Table 10. Each bank contains the fully functional filter, the complete actuator failure filters, and the complete sensor failure filters. In addition, Banks #2 through #6 contain three[19] spawned filters, each at a different effectiveness level. Notice the set of spawned filters for each of the five *spawning* banks is based on a different actuator failure. Each spawned filter is based on an effectiveness level $\epsilon_{i_a i_s}$, where i_a is the actuator index and i_s is the effectiveness level index. Since the first 12 filters are the same in Banks #2 through #6, changing banks is actually the method used to decide *which* spawned filters, if any, will be used.

Bank #1, in Table 10, clarifies how the MMAE works conceptually. The algorithm as implemented for this research only uses Banks #2 through #6, and all 15 filters are always used in the "Conventional MMAE Computations". However, if an actuator has not been declared failed, then only the first 12 filters really need to be used in the "Conventional MMAE Computations". This is similar to using Bank #1, except that additional computational loading is incurred during the "Conventional MMAE Computations". Further, this avoids banks that change the number of elemental filters running in parallel, as would be the case in going from Bank #1 to any other bank.

[19] *Three* spawned filters were chosen as a first-cut discretization. More filters results in finer discretization which provides a better estimate, but, the computational load is increased.

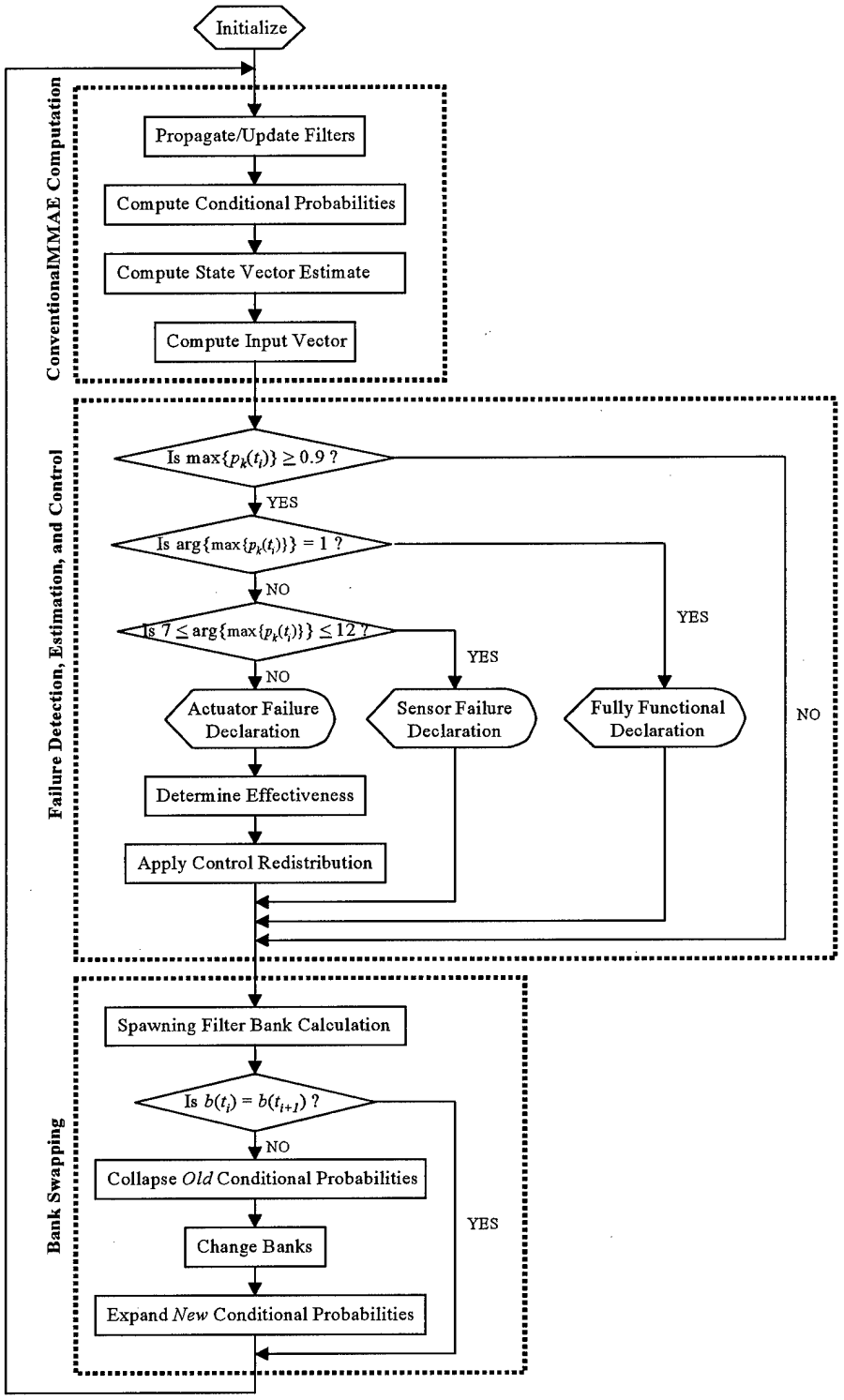


Figure 6. MMAE with Filter Spawning Flow Chart

Table 10. MMAE With Filter Spawning Bank Description

			Bank					
			1	2	3	4	5	6
	FF	1	FF	FF	FF	FF	FF	FF
		2	LS	LS	LS	LS	LS	LS
	Complete	3	RS	RS	RS	RS	RS	RS
	Actuator	4	LF	LF	LF	LF	LF	LF
F	Failure	5	RF	RF	RF	RF	RF	RF
i		6	R	R	R	R	R	R
l		7	AA	AA	AA	AA	AA	AA
t	Complete	8	PR	PR	PR	PR	PR	PR
e	Sensor	9	NA	NA	NA	NA	NA	NA
r	Failure	10	RR	RR	RR	RR	RR	RR
		11	YR	YR	YR	YR	YR	YR
		12	LA	LA	LA	LA	LA	LA
		13		$LS_{\epsilon=\epsilon_{21}}$	$RS_{\epsilon=\epsilon_{31}}$	$LF_{\epsilon=\epsilon_{41}}$	$RF_{\epsilon=\epsilon_{51}}$	$R_{\epsilon=\epsilon_{61}}$
	Spawned	14		$LS_{\epsilon=\epsilon_{22}}$	$RS_{\epsilon=\epsilon_{32}}$	$LF_{\epsilon=\epsilon_{42}}$	$RF_{\epsilon=\epsilon_{52}}$	$R_{\epsilon=\epsilon_{62}}$
		15		$LS_{\epsilon=\epsilon_{23}}$	$RS_{\epsilon=\epsilon_{33}}$	$LF_{\epsilon=\epsilon_{43}}$	$RF_{\epsilon=\epsilon_{53}}$	$R_{\epsilon=\epsilon_{63}}$

FullyFunctional (FF)	Left Stabilator (LS)	Right Stabilator (RS)
Left Flaperon (LF)	Right Flaperon (RF)	Rudder (R)
Angle of Attack (AA)	Pitch Rate (PR)	Normal Acceleration (NA)
Roll Rate (RR)	Yaw Rate (YR)	Lateral Acceleration (LA)

The alternate to using Bank #1 to exclude the spawned filters, as implemented in this research, is discussed in Section 4.4.2.

Recall that Section 3.5 detailed how each model is built. Each *unique* $\Phi, \mathbf{B}_{d-aug}$ and $\mathbf{F}_{sj} \cdot \mathbf{H}_{aug}$ matrix must be stored in memory. Although Table 10 lists $15 \cdot 5 = 75$ filters, the numbers of *unique* matrices are

Φ_{13x13}	$(\mathbf{B}_{d-aug})_{13x5}$	$(\mathbf{F}_{sj} \cdot \mathbf{H}_{aug})_{7x13}$
1	$1+5+15=21$	$1+6=7$

Because it was assumed that $\mathbf{A}_{fail} \approx \mathbf{A}$, all filters share the same Φ matrix. Because a different $\mathbf{F}_{ai}$ matrix for each actuator hypothesis is used to formulate $\mathbf{B}_{d-aug}$ in Equation (3.35), there are 21 unique $\mathbf{B}_{d-aug}$ matrices: one fully functional (shared by the sensor failure models) plus five – one for each of the five complete actuator failure models, and an additional 15 – one for each of the spawned filters for partial actuator failures. Because a different $\mathbf{F}_{sj}$ matrix for each sensor failure hypothesis is used, there are seven unique $\mathbf{F}_{sj} \cdot \mathbf{H}_{aug}$ matrices: one fully functional (shared by the complete and partial actuator failure models) and six more to handle each of the six sensor failure models. Notice the only additional load due to the spawned filters is 15 $\mathbf{B}_{d-aug}$ matrices of dimension $13x5$. Further, the seven $\mathbf{F}_{sj} \cdot \mathbf{H}_{aug}$ matrices can be implemented using one $\mathbf{H}_{aug}$, where the j^{th} row is zeroed on-line to perform the function of $\mathbf{F}_{sj}$. The noise strengths $\mathbf{Q}_d$ and $\mathbf{R}$, as defined in Table 4 and Table 7, respectively, are also stored in memory. Finally, if control redistribution is used, then the *packed, modified* control redistribution matrix, $\mathbf{D}_{a_{app}}(\gamma)$, must also be stored, where $\mathbf{D}_{a_{app}}(\gamma)$ is the $5x5$ matrix defined in Equation (3.55).

4.4.1.2 Initial Conditions. Initial conditions must be defined. The state vector estimate is initialized with trim condition values. Because MMAE uses only one bank at a time, a bank number must be specified. As an arbitrary choice, the MMAE starts in "Bank #2", or expressed as a function of time, "Bank $b(t_i)$", where $b(t_0) = 2$. Initial conditional probabilities must also be defined. The aircraft is assumed initially fully functional, where each failure filter's conditional probability will be equal to the lower bound 0.001 and the fully functional filter's conditional

probability is assigned such that the sum of the probabilities equals one. Thus,

$$p_k(t_0) = 0.001 \quad \text{for} \quad 2 \leq k \leq 15$$

$$p_1(t_0) = 1 - (14 \cdot 0.001) = 0.986$$

4.4.2 Conventional MMAE Computation

The "Conventional MMAE Computation" group performs the computations discussed in Section 2.3.2. Specifically, each elemental filter is propagated by Equations (2.11) and (2.12), repeated here as

$$\hat{\mathbf{x}}_{aug_k}(t_{i+1}^-) = \Phi \hat{\mathbf{x}}_{aug_k}(t_i^+) + \mathbf{B}_{d-aug}\mathbf{u}(t_i) \tag{4.7}$$

$$\mathbf{P}_k(t_{i+1}^-) = \Phi \mathbf{P}_k(t_i^+)\Phi^T + \mathbf{Q}_{d-aug} \tag{4.8}$$

Each elemental filter is updated by Equations (2.13) through (2.17), repeated here as

$$\mathbf{A}_k(t_i) = \mathbf{F}_{sj_k}\mathbf{H}_{aug}\mathbf{P}_k(t_i^-)\mathbf{H}_{aug}^T\mathbf{F}_{sj_k}^T + \mathbf{R} \tag{4.9}$$

$$\mathbf{K}_k(t_i) = \mathbf{P}_k(t_i^-)\mathbf{H}_{aug}^T\mathbf{F}_{sj_k}^T\mathbf{A}_k^{-1}(t_i) \tag{4.10}$$

$$\mathbf{r}_k(t_i) = \mathbf{z}_i - \mathbf{F}_{sj_k}\mathbf{H}_{aug}\hat{\mathbf{x}}_{aug_k}(t_i^-) \tag{4.11}$$

$$\hat{\mathbf{x}}_{aug_k}(t_i^+) = \hat{\mathbf{x}}_{aug_k}(t_i^-) + \mathbf{K}_k(t_i)\mathbf{r}_k(t_i) \tag{4.12}$$

$$\mathbf{P}_k(t_i^+) = \mathbf{P}_k(t_i^-) - \mathbf{K}_k(t_i)\mathbf{F}_{sj_k}\mathbf{H}_{aug}\mathbf{P}_k(t_i{}^-) \tag{4.13}$$

The system matrices $\Phi, \mathbf{B}_{d-aug}, \mathbf{F}_{sj_k}$, and $\mathbf{H}_{aug}$ are based on the current bank, Bank $b(t_i)$, and the noise strengths $\mathbf{Q}_d$ and $\mathbf{R}$ are stored in memory. The subscript k was dropped from matrices independent of the particular elemental filter, where the dependance of the matrices on $\mathbf{a}_k$ was described in Section 4.4.1.

From Section 2.3.2, the conditional probabilities, $p_k(t_i)$, are expressed in Equation (2.18), repeated here as

$$p_k(t_i) = Prob(\mathbf{a} = \mathbf{a}_k | \mathcal{Z}(t_i) = \mathcal{Z}_i) \tag{4.14}$$

The probabilities can be computed as [23] using Equation (2.19), repeated here as

$$p_k(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1})p_k(t_{i-1})}{\sum_{j=1}^{K} f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathcal{Z}_{i-1})p_j(t_{i-1})} \tag{4.15}$$

where, from Equations (2.21) and (2.22),

$$f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k,\mathcal{Z}_{i-1}) = \beta_k(t_i)e^{[-\frac{1}{2}\mathbf{r}_k(t_i)^T\mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i)]} \quad (4.16)$$

$$\beta_k(t_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\mathbf{A}_k(t_i)|^{\frac{1}{2}}} \quad (4.17)$$

To complete the conditional probability calculation, a probability lower bound of 0.001 is applied, as discussed in Section 2.4.1. The state vector estimate is obtained using the Bayesian blending method, expressed as

$$\hat{\mathbf{x}}^{Baysian}_{MMAE}(t_i^+) = \frac{\sum_{k\in\mathcal{B}}\hat{\mathbf{x}}_k(t_i^+)p_k(t_i)}{\sum_{k\in\mathcal{B}}p_k(t_i)} \quad (4.18)$$

where $\mathcal{B}$ contains all k such that $p_k(t_i) \geq 0.005$, as defined earlier in Equation (2.24) in Section 2.3.2.3, but with a blending lower bound of 0.005 applied as discussed in Section 2.4.1. The measurement vector, $\mathbf{z}(t_i)$, is reconstructed to reflect the state estimate [41,42], expressed as

$$\hat{\mathbf{z}}^{Baysian}_{MMAE}(t_i) = \mathbf{H}_{aug}\hat{\mathbf{x}}^{Baysian}_{MMAE}(t_i^+) \quad (4.19)$$

where $^{Baysian}_{MMAE}$ on $\hat{\mathbf{z}}$ denotes the measurement vector is based on a state vector estimate from an MMAE algorithm using Bayesian blending. The measurement vector estimate, $\hat{\mathbf{z}}^{Baysian}_{MMAE}(t_i)$, is used to replace the raw measurement $\mathbf{z}(t_i)$ as an input to the Block 40 FCS, thereby allowing for normal operation despite sensor failures. The Block 40 FCS then produces a control vector, $\mathbf{u}^{Baysian}_{MMAE}(t_i)$, where again, $^{Baysian}_{MMAE}$ on $\mathbf{u}$ denotes the control vector is based on a measurement vector using a state vector estimate from an MMAE algorithm using Bayesian blending.

4.4.3 Failure Detection, Estimation, and Control

The "Failure Detection, Estimation, and Control" group in Figure 6 applies MMAE to failure detection (using the conditional probabilities from models based on hypothesized failures). This group occurs in three stages: detection, estimation, and control. Failure detection uses a maximum a posteriori (MAP) approach. Estimation uses a Bayesian blending approach. Control is applied through MMAE with Control Redistribution.

4.4.3.1 Detection. Before detection can occur, the probabilities must be redefined to assure a proper comparison. Specifically, consider the actuator failure on which the spawned filters are based, with an actuator index of $i_a = b$. The conditional probability that the i_a^{th} actuator has

failed is dispersed among the i_a^{th} complete actuator failure filter and the spawned filters modeling partial failures of that actuator. As a remedy, the conditional probability of the i_a^{th} actuator filter is redefined:

$$p'_{i_a}(t_i) = p_{i_a}(t_i) + p_{13}(t_i) + p_{14}(t_i) + p_{15}(t_i) \tag{4.20}$$

where, from Table 10, i_a is the filter index for the complete actuator failure and 13 through 15 are the filter indices for the spawned filters. Thus, the sum of the conditional probabilities for the filters based on the i_a^{th} actuator for any effectiveness will be compared to the remaining filter's conditional probabilities. A more rigorous comparison would be to re-compute the conditional probabilities using Equation (2.19) *without considering the spawned filters* (by using Bank #1 in the "Conventional MMAE Computations"). While the formulation in Equation (4.20) is not as strict as re-computing the conditional probabilities without the spawned filters, the formulation in Equation (4.20) demonstrates successful results and saves computation time.

A failure is *detected* if the highest conditional probability exceeds some threshold, expressed as

$$1 \leq k \leq 12 \max\left[p'_k(t_i)\right] \geq \mathrm{P}_{threshold} \tag{4.21}$$

where $p'_k(t_i)$ denotes for $k = i_a$ the conditional probability $p'_{i_a}(t_i)$ is defined in Equation (4.20) and for $k \neq i_a$ the conditional probabilities are defined in Equation (4.15). The threshold, $\mathrm{P}_{threshold}$, for this research is 0.9; a lower threshold would detect failures faster but also increase false alarms [10, 11,31,32].

Equation (4.21) answers the question, "Has a failure occurred?"

- If no conditional probability exceeds this level, then no failure is assumed to have occurred and the failure declaration remains at the previous declaration, i.e., $\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i) = \hat{\mathbf{a}}^{MAP}_{MMAE}(t_{i-1})$.
- If a conditional probability does exceed this level, the failure dictated by Equation (4.22) below is declared to have occurred.

The failure is detected using a maximum a posteriori (MAP) approach, as defined in Equation (2.27), expressed as

$$\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i) = \mathbf{a}_j(t_i) \quad \text{where} \quad j = \arg\left\{1 \leq k \leq 12 \max\left[p'_k(t_i)\right]\right\} \tag{4.22}$$

where $\hat{\mathbf{a}}^{MAP}_{MMAE}$ is the failure estimate and $\mathbf{a}_j$ is the hypothesized failure for the j^{th} filter. The term $p'_k(t_i)$ is the conditional probability for the k^{th} filter, and for $k = i_a$, the conditional probability $p'_{i_a}(t_i)$ is used as defined in Equation (4.20).

Equation (4.22) answers the question, "What failure has occurred?"

- If $j = 1$ in Equation (4.22), then $\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i)$ corresponds to a fully functional aircraft occurring.
- If $7 \leq j \leq 12$, then $\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i)$ corresponds to a sensor failure occurring.
- Otherwise, i.e., $2 \leq j \leq 6$, $\hat{\mathbf{a}}^{MAP}_{MMAE}(t_i)$ corresponds to an actuator failure occurring.

4.4.3.2 ***Estimation.*** After detecting a failure, the extent of the failure is *estimated* in terms of effectiveness. For this research, only complete sensor failures are considered, while complete and partial actuator failures are considered. As a result, estimating the effectiveness for a sensor is accomplished using *implicit estimation*; that is, a separate estimation algorithm is not required because the only hypothesized effectiveness is $\epsilon = 0$. Therefore, for detected sensor failures, the effectiveness estimate is $\hat{\epsilon} = 0$. On the other hand, actuator failures have more than one effectiveness hypothesized (through the use of filter spawning), requiring *explicit estimation*, meaning estimation must be accomplished separate from, and subsequent to, detection.

The explicit estimation action is shown on the flow chart in Figure 6 as the block, "Determine Effectiveness". Recall that filter spawning is used to enhance estimation, resulting in the addition of hypothesized effectiveness values[20] $\epsilon_{i_a i_s}$ for $i_s = 1, 2, 3$. Given that the failure index is j, as defined in Equation (4.22), then the actuator index, i_a, is $i_a = j$ using the index definitions in Table 10. The hypothesized effectiveness values are then

$$\epsilon_{j0} = 0,\ \epsilon_{j1},\ \epsilon_{j2},\ \epsilon_{j3},\ \epsilon_{ff} = 1 \tag{4.23}$$

where ϵ_{j0} indicates the complete actuator failure filter's hypothesized effectiveness value; $\epsilon_{j1}, \epsilon_{j2}$, and ϵ_{j3} indicate the spawned filters' hypothesized effectiveness values; and ϵ_{ff} indicates the fully functional filter's hypothesized effectiveness value of one.

[20]If the current spawned filters do not correspond to the currently declared actuator failure, then the algorithm must delay using the spawned filters until the new ones are brought on-line. Because of the bank swapping logic used in Section 4.4.4, this is usually not the case.

The effectiveness is *estimated* using the Bayesian blending approach using Equation (4.6). Using the effectiveness values in Equation (4.23), Equation (4.6) can be expressed as

$$\hat{\epsilon}(t_i) = \frac{p_j(t_i) + \epsilon_{j1}p_{13}(t_i) + \epsilon_{j2}p_{14}(t_i) + \epsilon_{j3}p_{15}(t_i)}{p_j(t_i) + p_{13}(t_i) + p_{14}(t_i) + p_{15}(t_i) + p_1(t_i)} \tag{4.24}$$

where $\hat{\epsilon}$ is the effectiveness estimate.

Equation (4.24) answers the question, "To what extent has an actuator failure occurred?" The failure is assumed to have occurred to the extent dictated by the effectiveness estimate, $\hat{\epsilon}(t_i)$, as computed in Equation (4.24).

4.4.3.3 Control. Recall that, in the "Conventional MMAE Computation" group, the input vector was computed by the Block 40 FCS. Recall from Section 3.6 that, if an actuator has failed, Control Redistribution is used to *redistribute* control away from the failed actuator to the unfailed actuators. The redistributed control vector,$\mathbf{u}_r$, is defined in Equation (3.41) as

$$\mathbf{u}_r(t_i) = \mathbf{D}_{ai}\mathbf{u}_{MMAE}^{Bayesian}(t_i) \tag{4.25}$$

where, repeated from Equation (3.45),

$$\mathbf{D}_{ai} = (\mathbf{B}\mathbf{F}_{ai})^{+}\mathbf{B} \tag{4.26}$$

and $\mathbf{u}_{MMAE}^{Bayesian}(t_i)$ was generated by the Block 40 FCS based on an input of $\hat{\mathbf{z}}_{MMAE}^{Baysian}(t_i)$. The redistribution matrix is pre-computed for this application, as defined in Equation (3.48), and is stored in memory during initialization. Further, if a true inverse exists for $(\mathbf{B}\mathbf{F}_{ai})^{+}$, as in partial failures, *modified* Control Redistribution is used, defined in Equation (3.50) as

$$\mathbf{u}_{r_{mod}} = \mathbf{u}|_{\epsilon=1} + (1-\epsilon_0)\,\mathbf{u}_r|_{\epsilon=0} \tag{4.27}$$

where the effectiveness estimate, $\hat{\epsilon}$, defined in Equation (4.24) can be used as the needed ϵ_0. Rather than use $\hat{\epsilon}$ directly, Chapter 5 will *refine* the effectiveness estimate using the empirically observed relationship between the estimate of the effectiveness and the true effectiveness. Section 5.6 will present the decision to use such refined parameter estimates for both conventional Control Redistribution (for complete failures) and modified Control Redistribution (for partial failures).

4.4.4 *Bank Swapping*

Due to the hierarchical structure of the chosen implementation, changing the spawned filters requires *bank swapping*, shown in the "Bank Swapping" group in Figure 6. Bank swapping changes the bank number on which the MMAE algorithm's filter models are based. (Recall that the "Conventional MMAE Computation" group depended on the current bank number.) This section details the bank number calculation and conditional probability treatment during the bank change.

4.4.4.1 ***Bank Calculation.*** The index corresponding to the *actuator* filter with highest probability at time t_i is used to decide which bank will be used at the next time step, t_{i+1}. Thus, the bank number, b, at time t_{i+1} is expressed as

$$b\left(t_{i+1}\right) = \arg\left\{2 \leq k \leq 6\max\left[p'_k\left(t_i\right)\right]\right\} \tag{4.28}$$

where $p'_k\left(t_i\right)$ denotes the conditional probability $p'_{i_a}\left(t_i\right)$ as defined in Equation (4.20) for $k = i_a$, and the conditional probabilities as defined in Equation (4.15) for $k \neq i_a$. Notice that the maximum is taken over the indices corresponding to the actuators. Now, the spawned filters in the MMAE bank used at the next time step, Bank $b\left(t_{i+1}\right)$, will correspond to the actuator with the highest probability at the current time step.

Equation (4.28) answers the question, "Which partial actuator filters should be spawned next?" The partial filters contained in Bank $b\left(t_{i+1}\right)$, where $b\left(t_{i+1}\right)$ is calculated in Equation (4.28), should be used next.

Because the banks only differ in the spawned filters, bank swapping simply chooses which spawned filters are used. For this implementation, the spawned filters are *always* used. As an alternative, the spawned filters could be used only if the largest $p'_k\left(t_i\right)$ corresponds to an actuator failure. If the largest $p'_k\left(t_i\right)$ is for a fully functional aircraft or sensor failure, then the spawned filters should not be used. (This is equivalent to using Bank #1.)

4.4.4.2 ***Swapping Conditional Probabilities.*** Before a new bank can be used, initial conditional probabilities of the elemental filters within the new bank must be defined. To do this,

first notice in Table 10 that all banks share the same first 12 filters. The spawned filters, however, differ from bank to bank. Thus, simply assigning the conditional probabilities of the old bank to the new bank is not valid.

The spawned filters' conditional probabilities must be "collapsed" into the complete failure actuator filter on which the spawned filters are based. A rigorous "collapse" would be to re-compute the conditional probabilities using Equation (2.19) *without considering the spawned filters.* However, the formulation in Equation (4.20) is used because it saves computation time and the probabilities only serve as initial conditions for the new bank. The probabilities will disperse into the correct channels after a few sample periods.

To enhance this dispersion, the probability associated with the actuator filter that triggered a bank swap is "expanded" to the spawned filters. Rather than simply assign lower bound probabilities to the spawned filters and the remainder of the complete-actuator-failure filter's original probability to itself, the probability is divided equally among the complete-failure filter and the spawned filters. Thus, the spawned filters are lifted further from zero, allowing the probabilities to increase faster[21]. Also there is no *prebiasing* of relative probabilities among the complete-failure filter and the three spawned filters, since all really are equally likely, based on the information available at this time.

To summarize, if a change in spawned filters in declared, all of the old spawned filters' probabilities are collapsed into the associated complete-failure filter probability, and then the newly spawned filters and the newly declared complete-failure filter equally share the probability that had originally been in that complete-failure filter. The probability assignment "collapse" and "expansion" can be summarized as

Condition	**Assignment**
$1 \leq k \leq 12$ except $k = b(t_i) - 1$ and $k = b(t_{i+1})$	$p_k(t_{i+1}) = p_k(t_i)$
$k = b(t_i)$	$p_k(t_{i+1}) = p_k(t_i) + p_{13}(t_i) + p_{14}(t_i) + p_{15}(t_i)$
$k = b(t_{i+1})$ and $13 \leq k \leq 15$	$p_k(t_{i+1}) = \frac{p_{b(t_{i+1})}(t_i)}{4}$

[21] Because of the recursive nature of the probability calculation, the initial condition affects the time required for a filter to increase its probability. See Section 2.4.1 for the discussion on the *effects* of lower bounding.

4.4.5 Design Modification Options

Presented in Figure 6 and described throughout this section is one possible implementation for Filter Spawning. While other possibilities exist, this thesis will not explore the effects of using different implementations. In particular, the three questions posed throughout this section may be answered differently:

1. *Equation (4.21) answers the question, "Has a failure occurred?"* The probability threshold may be set differently than 0.9. Increasing this threshold may caused failures to be missed (i.e., the failure goes unnoticed), while decreasing this threshold may increase false alarms (i.e., failures are declared when either no failure exists or it is the incorrect failure declaration). Further, techniques discussed in Section 2.4 may be employed; namely, the blending lower bound may be set differently than 0.005, the scalar penalty may be adjusted from the current value $-\frac{1}{2}$, the conditional probabilities may be smoothed (through windowing techniques), or the conditional probabilities may be quantized.

2. *Equation (4.24) answers the question, "To what extent has an actuator failure occurred?"* The blending lower bound used to form the effectiveness estimate may be set differently than 0.005. Further, computing the conditional probabilities used in the estimate may be calculated using Equation (2.19) rather than using the $p'_k(t_i)$ calculated in Equation (4.20) for $k = i_a$ and calculated in Equation (4.15) for $k \neq i_a$. Specifically, Section 5.6 of the next chapter presents a *refined* estimate of the extent of an actuator failure, for use by the Control Redistribution algorithm.

3. *Equation (4.28) answers the question, "Which partial actuator filters should be spawned next?"* Selecting the next bank and deciding when to use spawned filters can be modified.

4.5 Chapter Summary

This chapter started by reviewing in detail the necessary models used in the MMAE algorithm. Filter Spawning was discussed conceptually to motivate its use and to give an overview of

its enhancement to estimation. Finally, Filter Spawning was detailed as it is implemented for this research. Each phase of the flow chart in Figure 6 was discussed: "Initialization", "Conventional MMAE Computation", "Detection, Estimation, and Control", and "Bank Swapping". The reader should have sufficient knowledge to duplicate the use of Filter Spawning as implemented in this research, as well as be able to understand the impact of the results presented in the following chapter.

Chapter 5 - Simulation Results

5.1 Chapter Overview

This chapter presents the simulation results of the MMAE/CR algorithm with Filter Spawning to address partial failures of actuators. Discretization sets for the spawned filters are chosen in Section 5.2. The conditional probabilities at each sample time are plotted in Section 5.3. Using this information, a blended estimate of the effectiveness of the partially failed actuator is found at each sample time in Section 5.4. Averaging this estimate over time, a parameter estimate for each true parameter value is found in Section 5.5. Using the knowledge of this relationship, the estimate is refined in Section 5.6 before using it for Control Redistribution (CR). Finally, Section 5.7 summarizes the design and implementation processes.

5.2 Discretization Sets

In Multiple Model Adaptive Estimation (MMAE) with Filter Spawning, filters are spawned with an effectiveness value $\epsilon_{i_a i_s}$, where i_a is the actuator failure index and i_s is the spawned filter's assumed effectiveness index. For this research, three spawned filters are used, i.e., $i_s = 1, 2, 3$. The subscript i_a permits the use of different discretization levels for each actuator. For simplicity, all actuators will use the same level of discretization in this research; that is, $\epsilon_{i_a 1}, \epsilon_{i_a 2}$, and $\epsilon_{i_a 3}$ are the same for all i_a.

Four sets of discretizations are considered, as shown in Table 11. Set #1 was chosen to be consistent with prior research. Clark [8] estimated the effectiveness by blending the fully functional filter's estimate with the complete actuator failure filter's estimate, using the respective conditional probabilities as the blending weighting. Further, the estimate was *quantized* to one of the following effectiveness values: 10%, 25%, and 50%. This was done because it was thought that estimating a *small* effectiveness value (i.e., a high percentage failure) would be easier than estimating a *large* effectiveness value. Set #2 was chosen after initial studies of Set #1 showed that $\epsilon_{i_a 1} = 10\%$ was too close to $\epsilon = 0\%$ as hypothesized by the complete actuator failure filter. (Through the results

presented in this chapter, the validity of this statement will be clear.) Thus, Set #2 provides a discretization that favors estimating *small* effectiveness values over *large* effectiveness values. In contrast, Set #4 was chosen to provide a discretization that favors estimating *large* effectiveness values over *small* effectiveness values. Set #3 was chosen to be impartial to small and large effectiveness values.

Table 11. Discretization Sets

	Set #1	Set #2	Set #3	Set #4
$\epsilon_{i_a}1$	10%	20%	25%	40%
$\epsilon_{i_a}2$	25%	40%	50%	60%
$\epsilon_{i_a}3$	50%	60%	75%	80%

5.3 Probability Plot

5.3.1 Description

The simulation starts at time equal to zero, with a fully functional aircraft in steady level flight at 0.4 Mach at 20,000 feet. At one second, a failure is introduced into the truth model. The failure is maintained for the duration of the simulation, or seven additional seconds. While only actuator failures will be shown, complete sensor failures were considered to *validate the design.* That is, the introduction of filter spawning is shown not to affect the detection of complete sensor failures as demonstrated in previous research [8, 18, 41, 42].

The "probability plot" is a plot of the conditional probability for each filter over time. Ten Monte Carlo runs were performed for each simulation. The mean is plotted as solid line and (mean $\pm$ 1σ) bounds are plotted as dotted lines. The conditional probabilities generated by the MMAE with Filter Spawning algorithm at each time sample were written to a data file. The only post-processing was computing the mean and standard deviation over the 10 Monte Carlo runs.

There are 15 filters in the MMAE algorithm (see Table 10 on page 59). A "probability plot" for a 20% left stabilator failure using the discretization set #1 is shown in Figure 7. Of most significance are the top two plots (probabilities for fully functional aircraft and complete left stabilator failure hypotheses) and the last three plots (probabilities for the three spawned filters). Notice that the

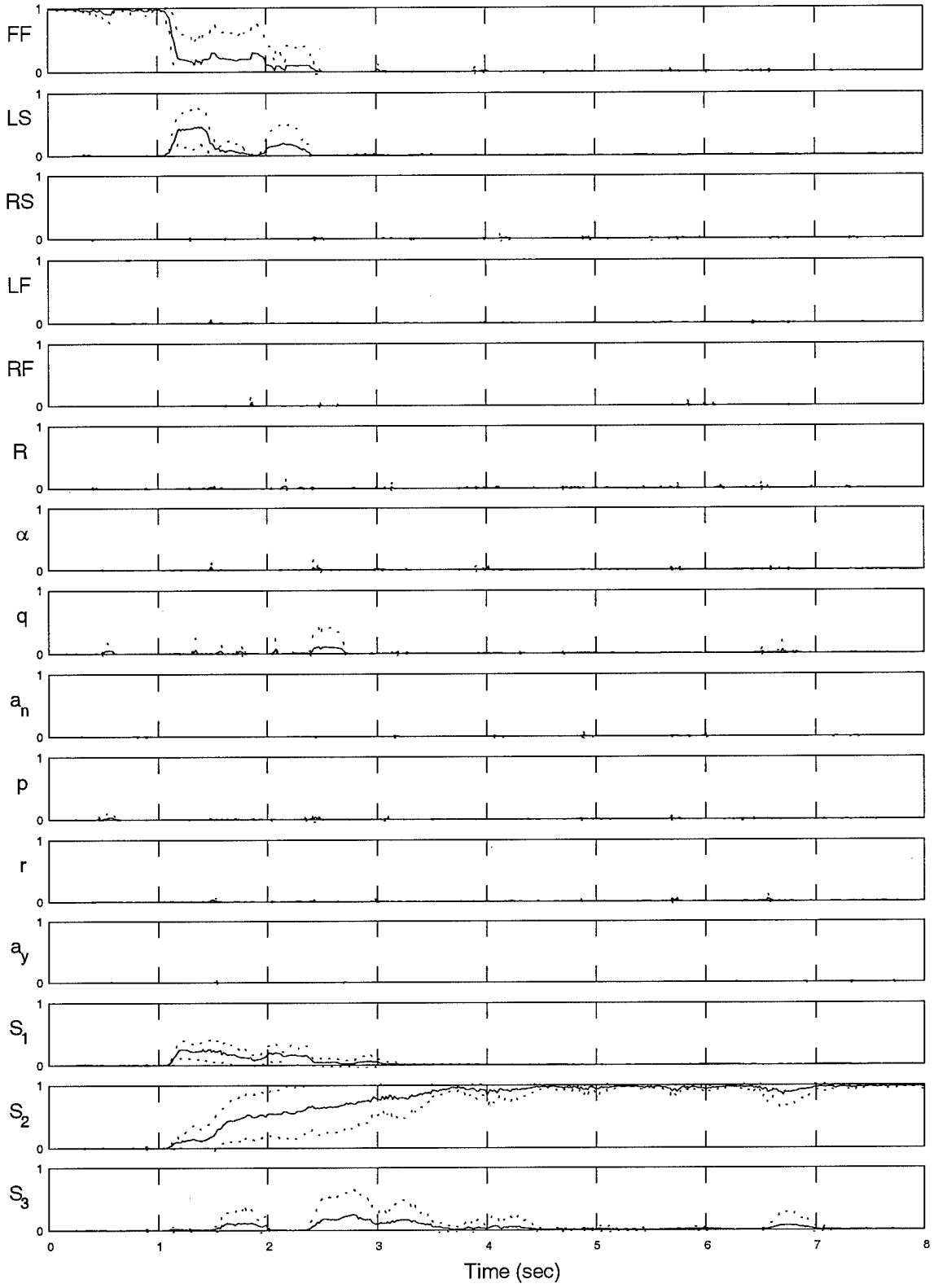


Figure 7. Probability Plot (All Channels): Left Stabilator Failure, $\epsilon_{true} = 20\%$, Spawned Filter Discretization $10\%, 25\%, 50\%$

channels with *wrong* hypotheses remain near zero during most of the simulation. These filters' probabilities provide little information; they verify that *wrong* filters do not obtain high probabilities. As a result, the "probability plots" throughout the remainder of this section do not show these filters' conditional probabilities. (While the probabilities of *wrong* filters are not shown, they were verified to be near zero during most of the simulation for every case considered.) Explicitly, the remaining "probability plots" show the conditional probabilities for the fully functional filter, the complete actuator failure filter, and the three spawned filters.

5.3.2 Results

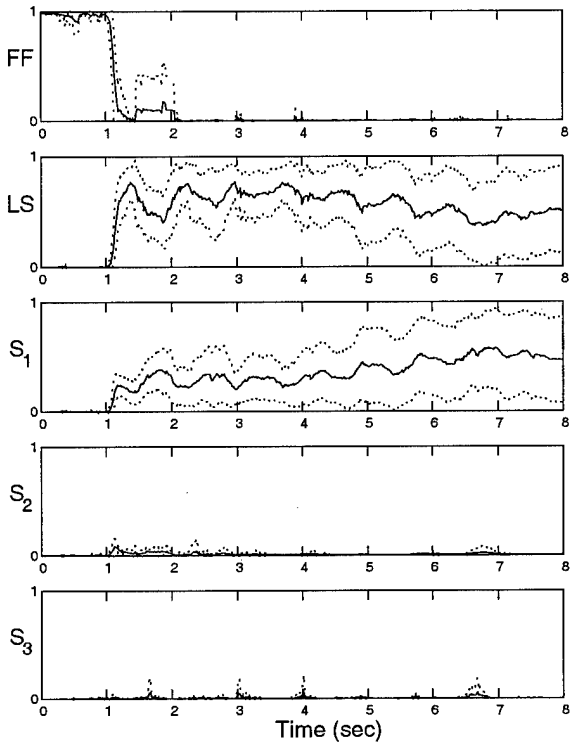
Probability plots are formed varying the discretization set, the actuator, and the effectiveness. Eleven effectiveness values are considered:

$$\epsilon_{true} = 0, 10, 20, \cdots, 100 \tag{5.1}$$

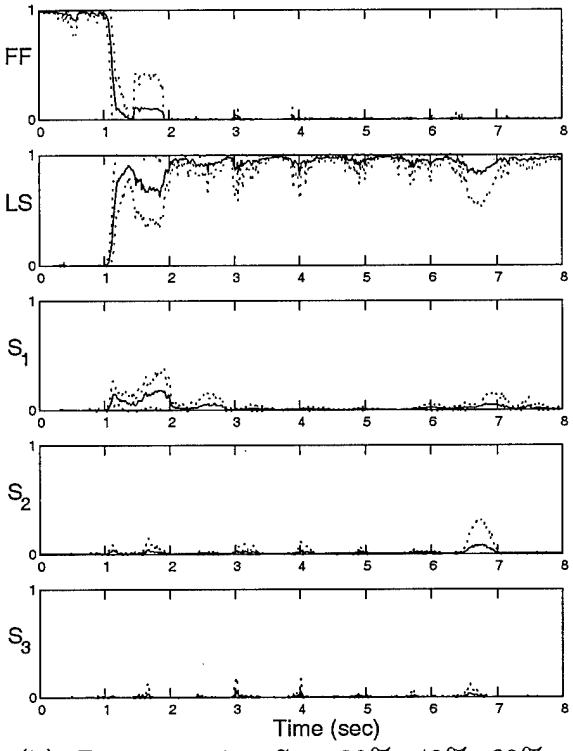
Notice that the fully functional and *complete* actuator failure cases are included. Each actuator failure at a given effectiveness (for each of the eleven effectiveness values given in Equation (5.1)) is shown while the discretization varies among the four sets of Table 10 within subplots (a) through (d) of each figure. Presented here are the probability plots for the *left stabilator*. The right stabilator, left flaperon, right flaperon, and rudder probability plots are shown in Appendix B. The corresponding figure number and page number for each effectiveness value of the left stabilator failure are given in Table 12.

Table 12. Figure Numbers for Left Stabilator Failure Probability Plots

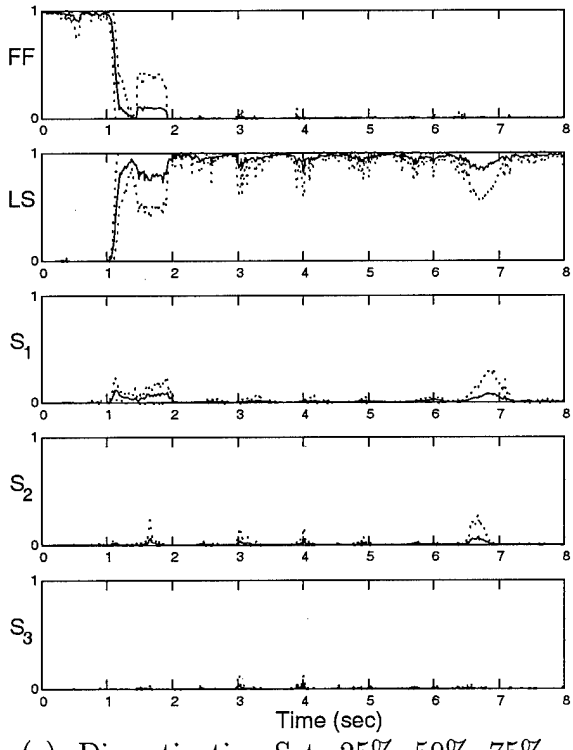
Figure	Left Stabilator Failure Effectiveness, ϵ_{true}	Page
8	0%	74
9	10%	75
10	20%	76
11	30%	77
12	40%	78
13	50%	79
14	60%	80
15	70%	81
16	80%	82
17	90%	83
18	100%	84



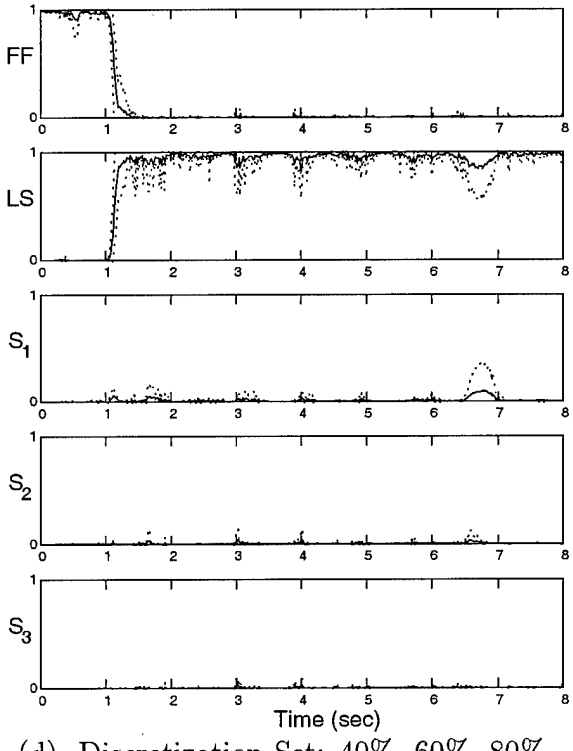
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

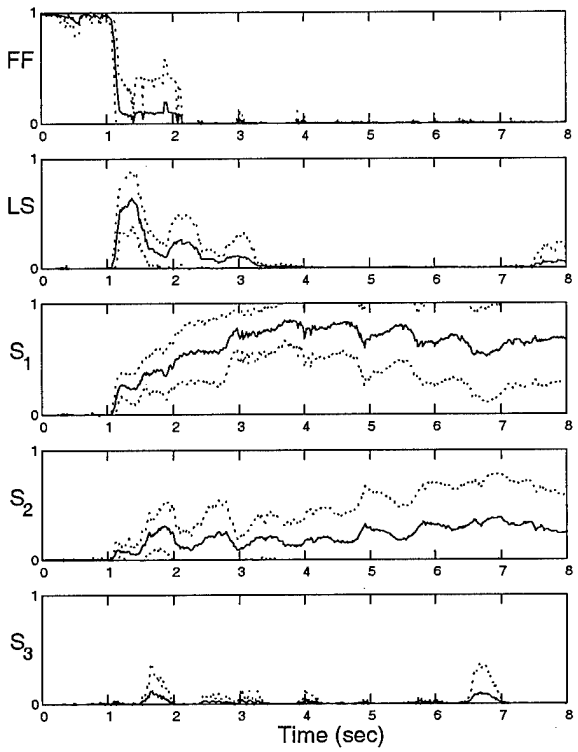


(c) Discretization Set: 25%, 50%, 75%

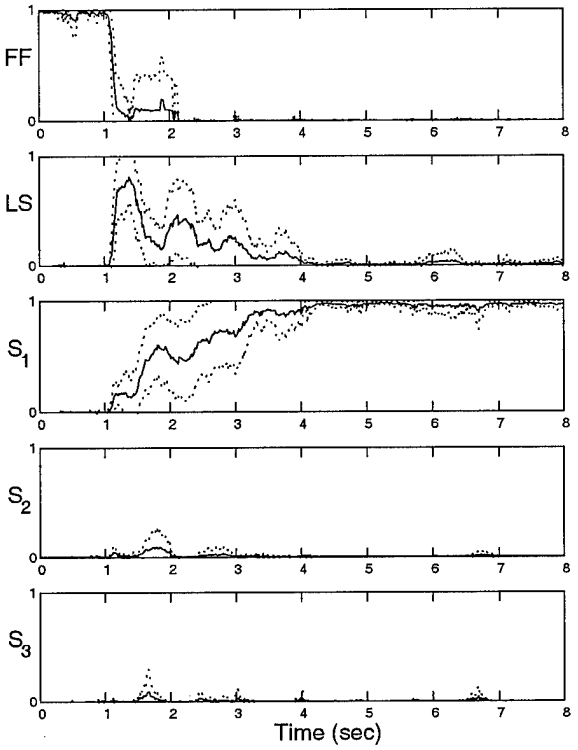


(d) Discretization Set: 40%, 60%, 80%

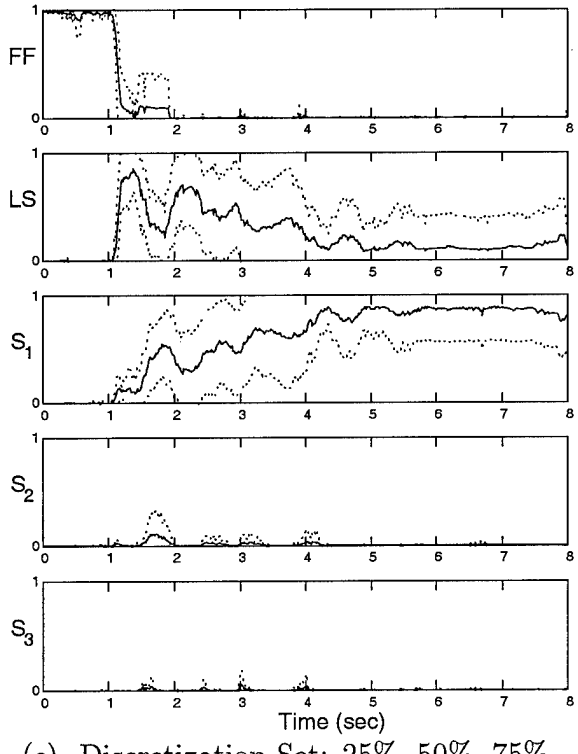
Figure 8. Probability Plot: Left Stabilator Failure, $\epsilon = 0\%$



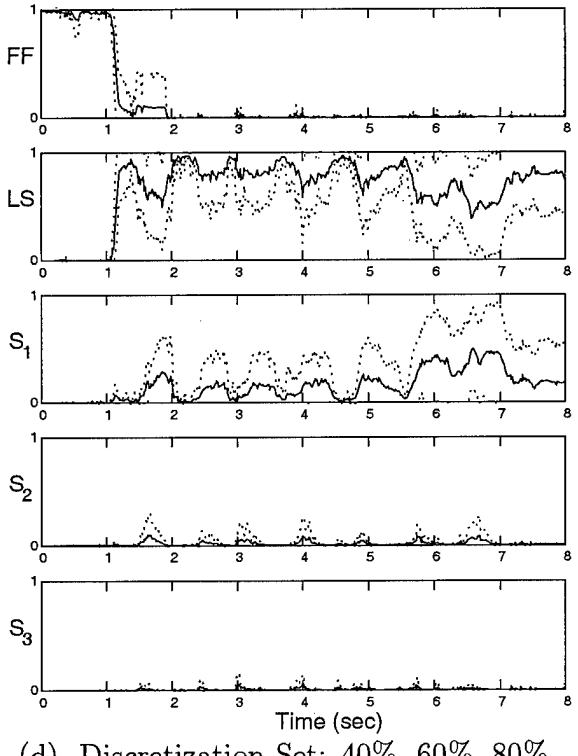
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

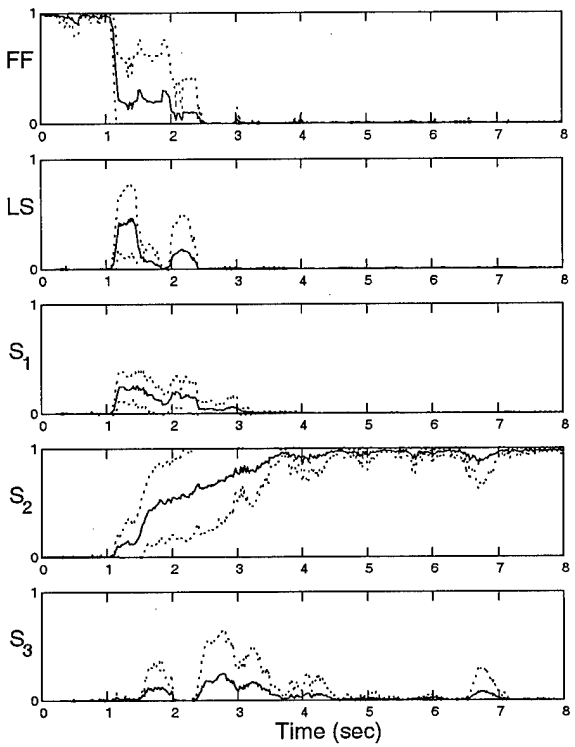


(c) Discretization Set: 25%, 50%, 75%

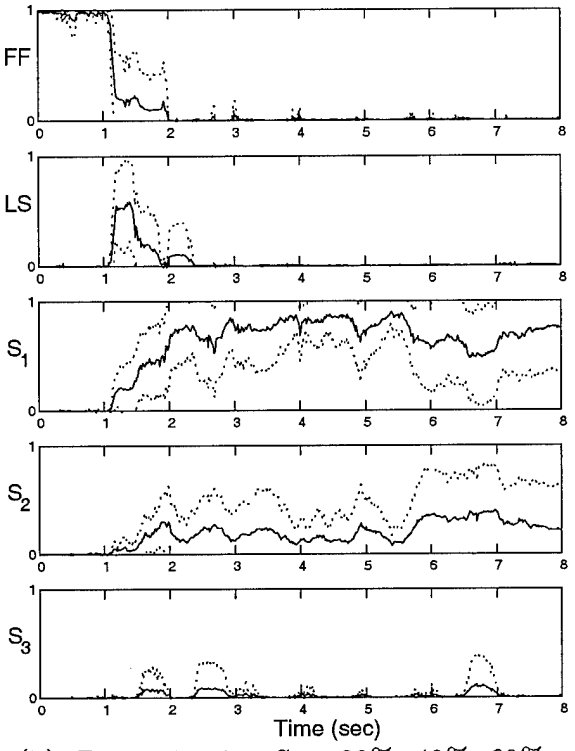


(d) Discretization Set: 40%, 60%, 80%

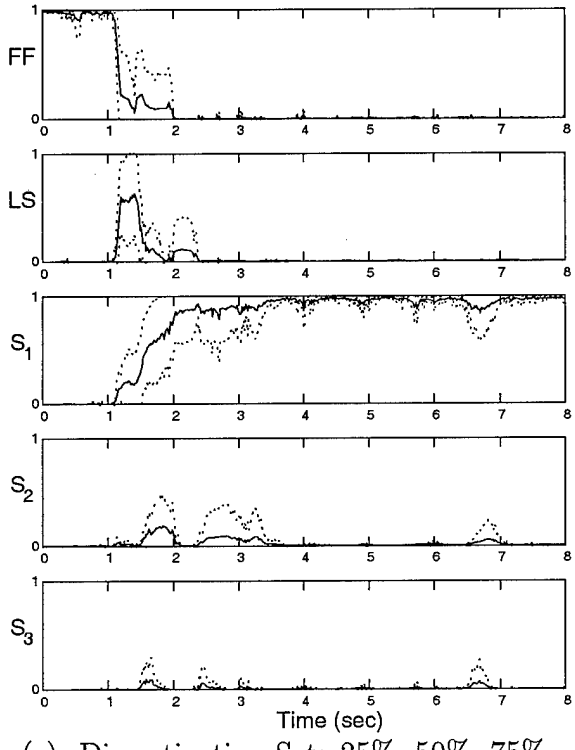
Figure 9. Probability Plot: Left Stabilator Failure, $\epsilon = 10\%$



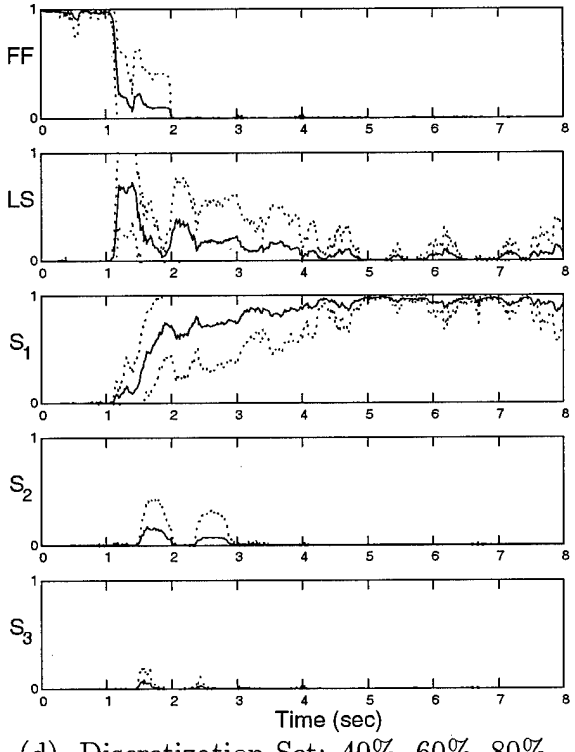
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

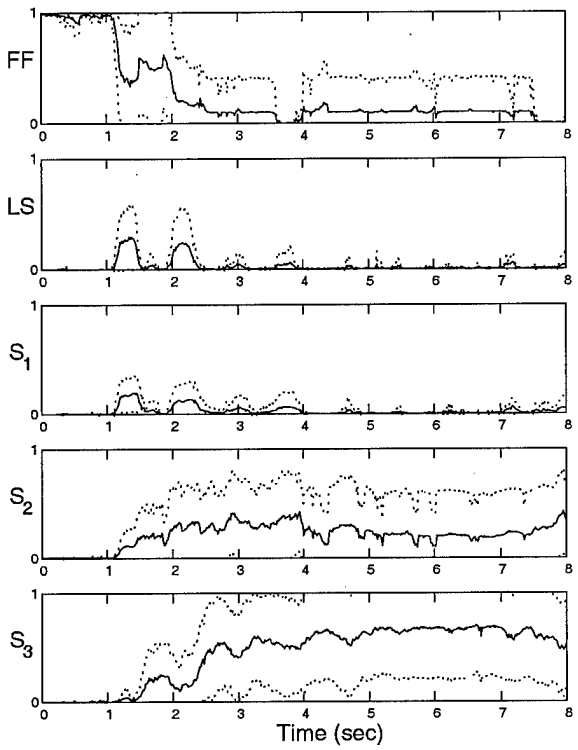


(c) Discretization Set: 25%, 50%, 75%

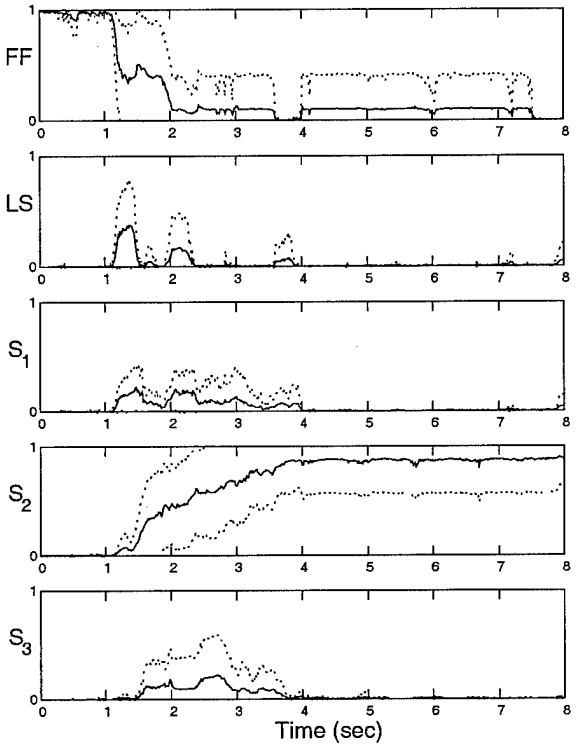


(d) Discretization Set: 40%, 60%, 80%

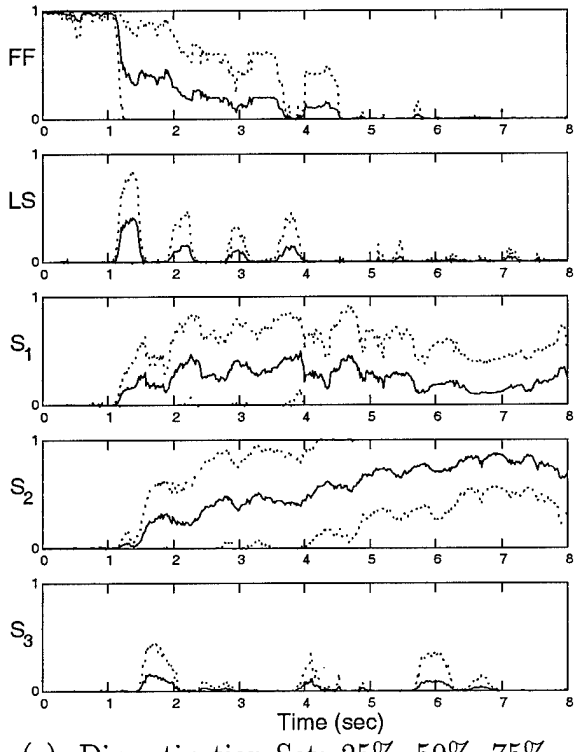
Figure 10. Probability Plot: Left Stabilator Failure, $\epsilon = 20\%$



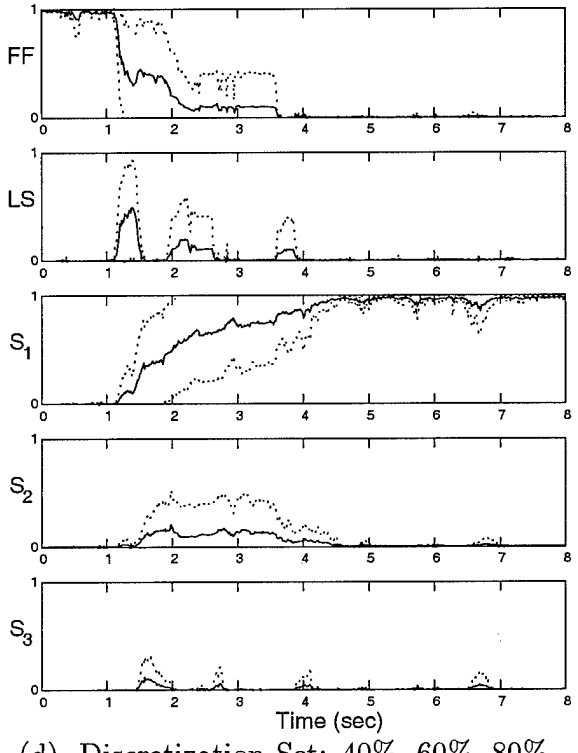
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

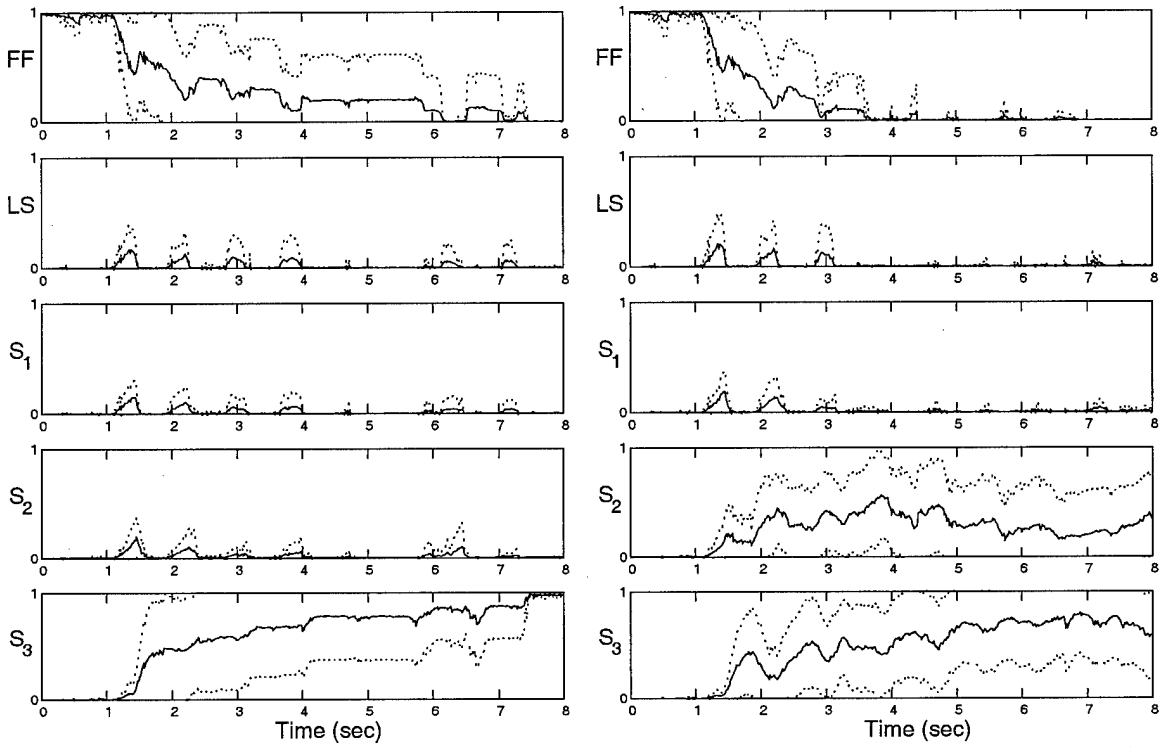


(c) Discretization Set: 25%, 50%, 75%



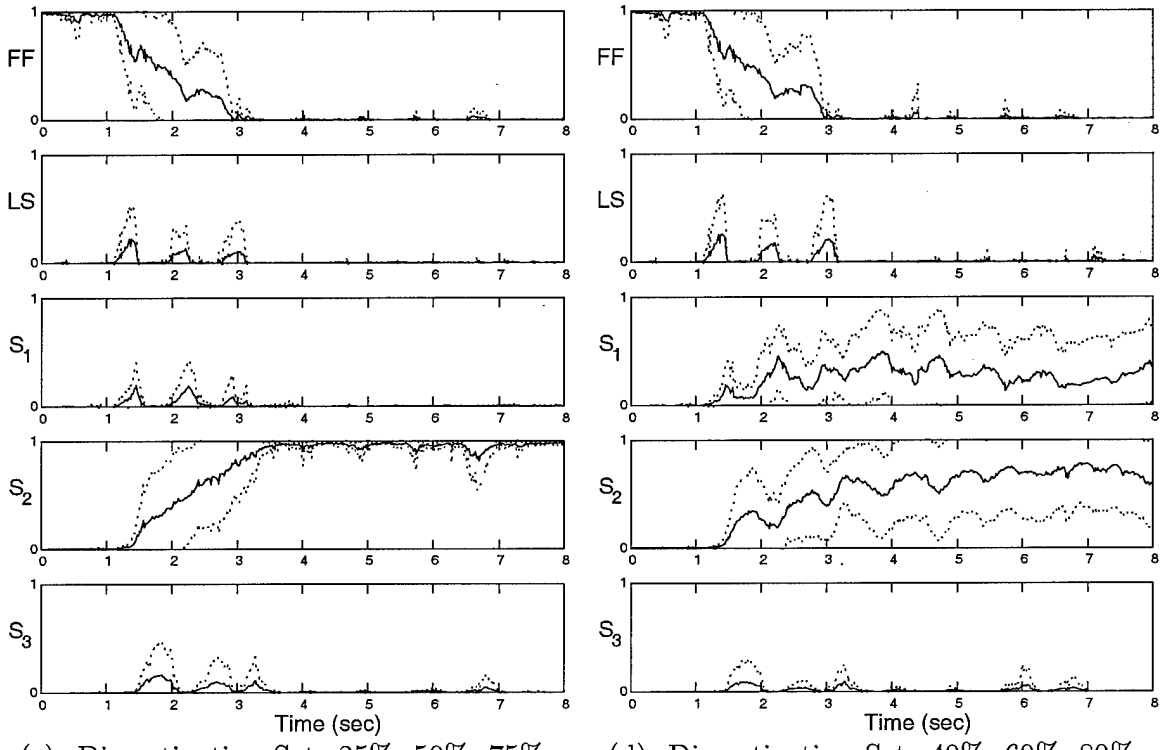
(d) Discretization Set: 40%, 60%, 80%

Figure 11. Probability Plot: Left Stabilator Failure, $\epsilon = 30\%$



(a) Discretization Set: 10%, 25%, 50%

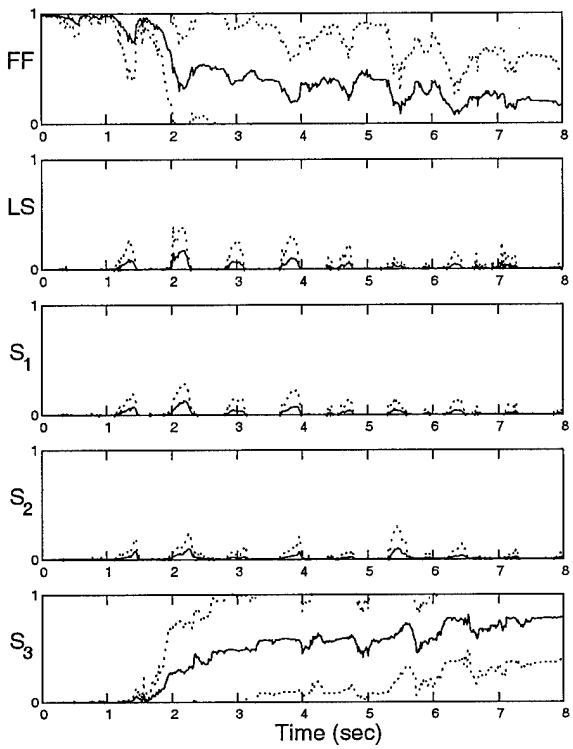
(b) Discretization Set: 20%, 40%, 60%



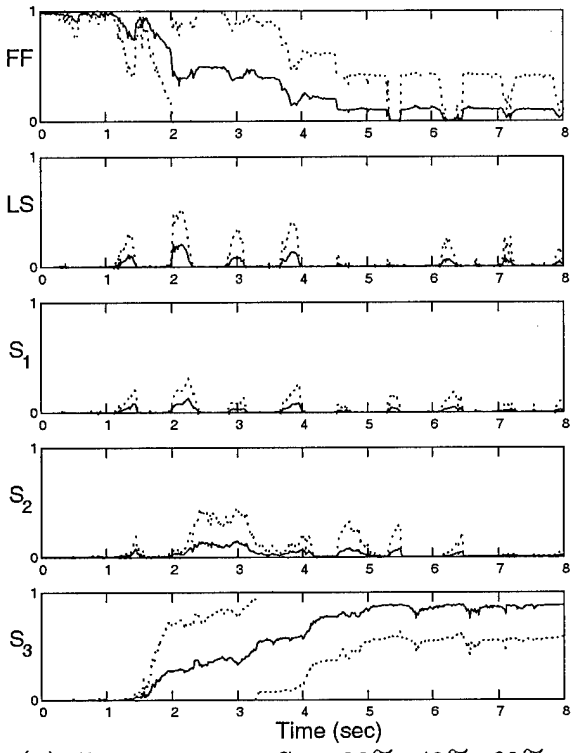
(c) Discretization Set: 25%, 50%, 75%

(d) Discretization Set: 40%, 60%, 80%

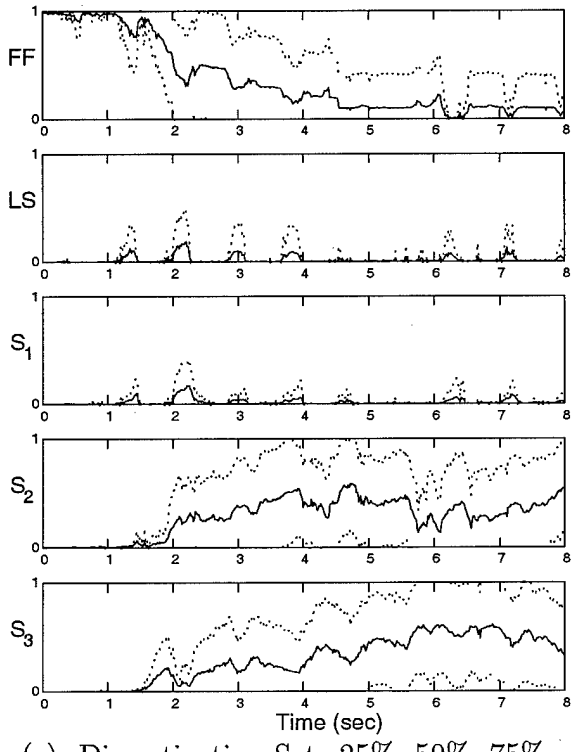
Figure 12. Probability Plot: Left Stabilator Failure, $\epsilon = 40\%$



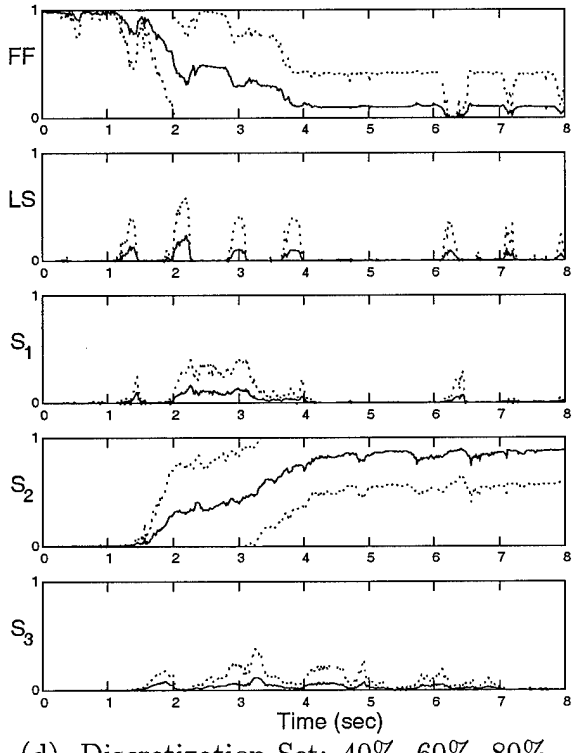
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

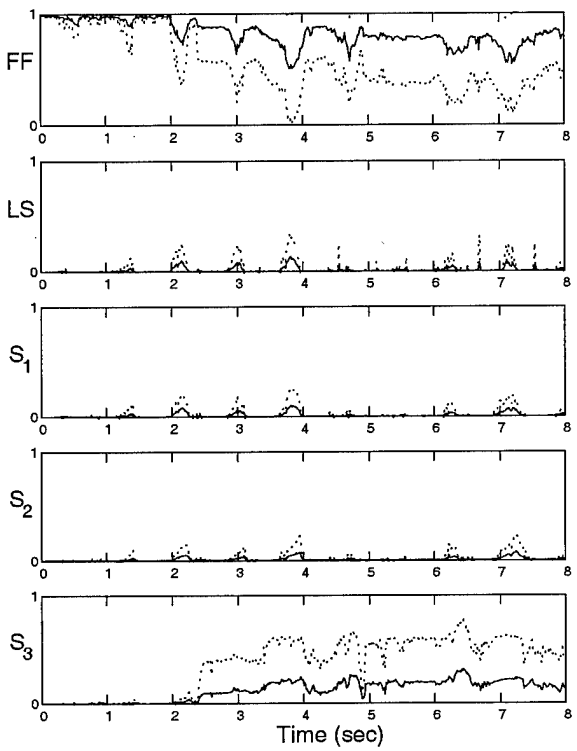


(c) Discretization Set: 25%, 50%, 75%

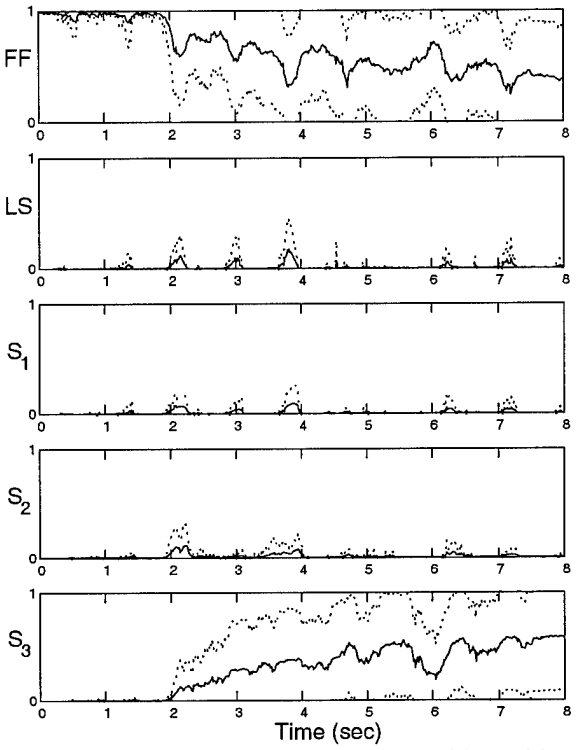


(d) Discretization Set: 40%, 60%, 80%

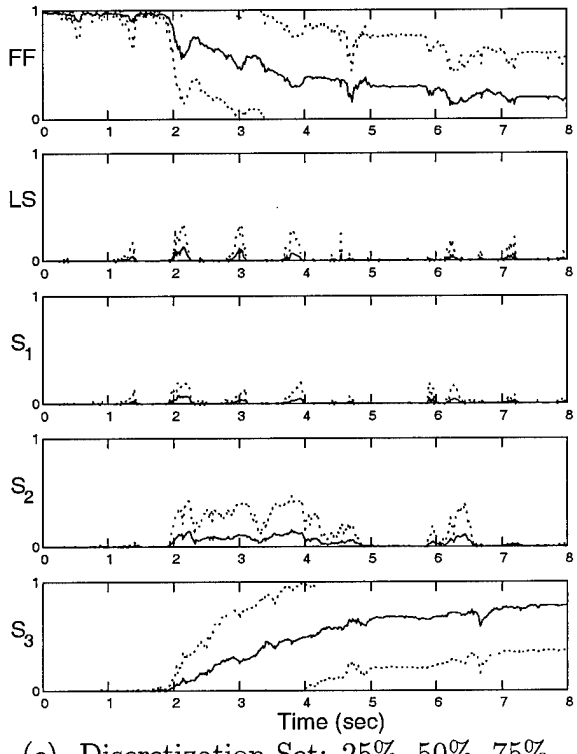
Figure 13. Probability Plot: Left Stabilator Failure, $\epsilon = 50\%$



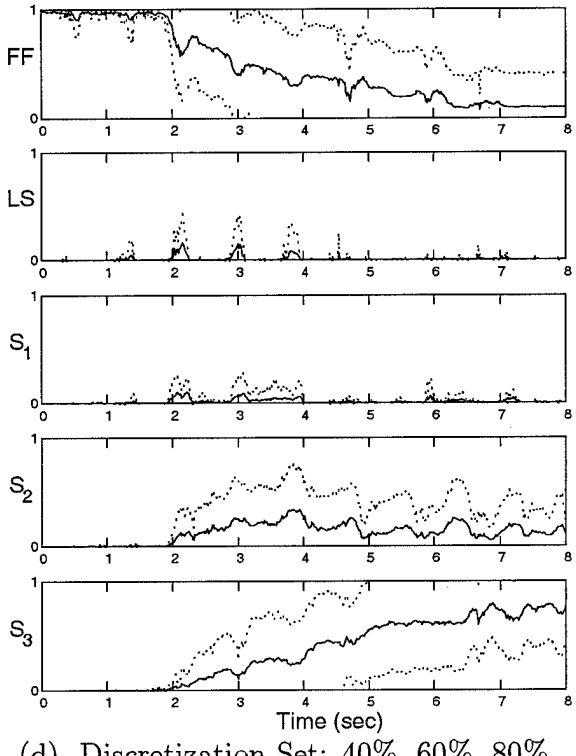
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

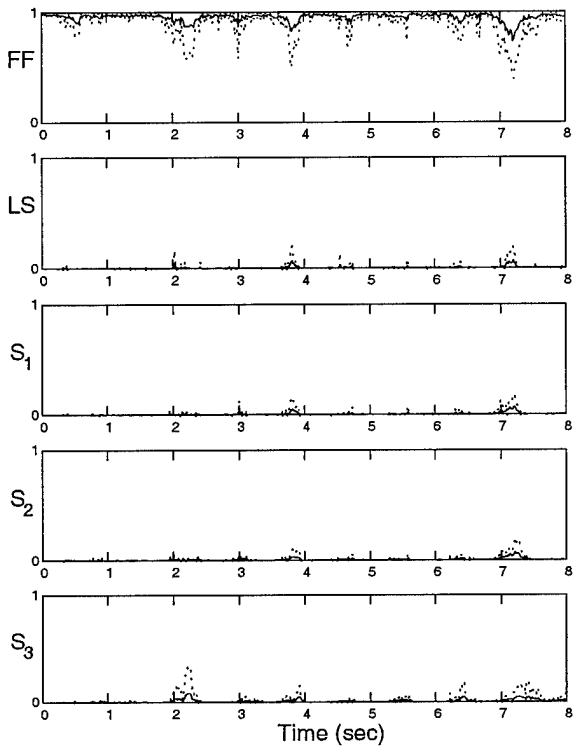


(c) Discretization Set: 25%, 50%, 75%

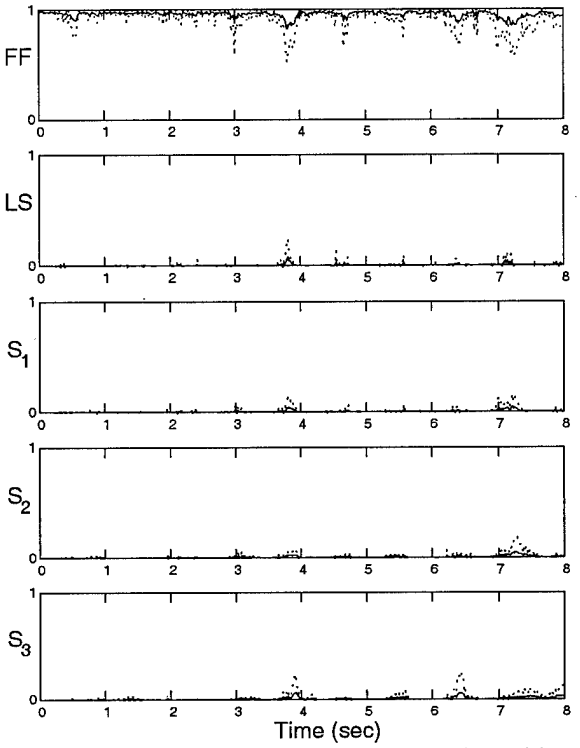


(d) Discretization Set: 40%, 60%, 80%

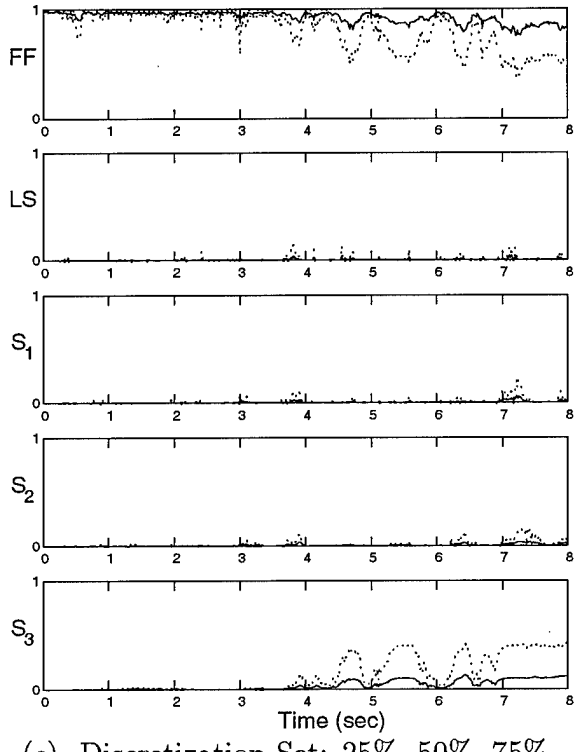
Figure 14. Probability Plot: Left Stabilator Failure, $\epsilon = 60\%$



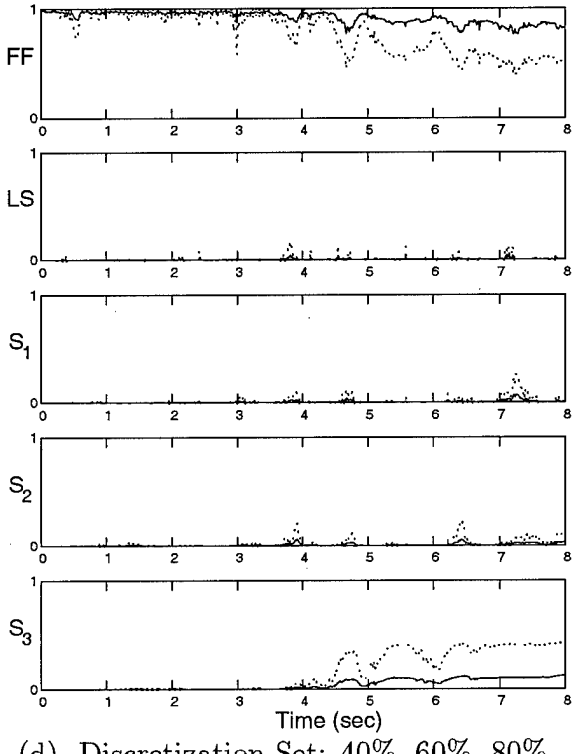
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

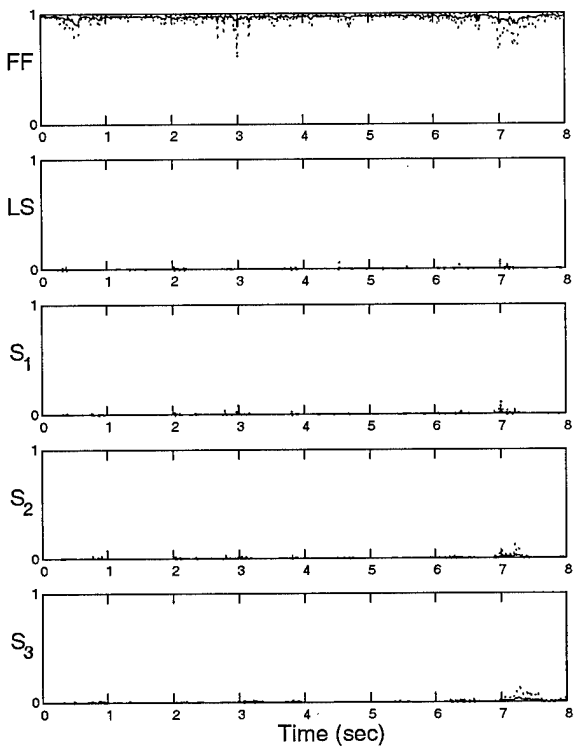


(c) Discretization Set: 25%, 50%, 75%

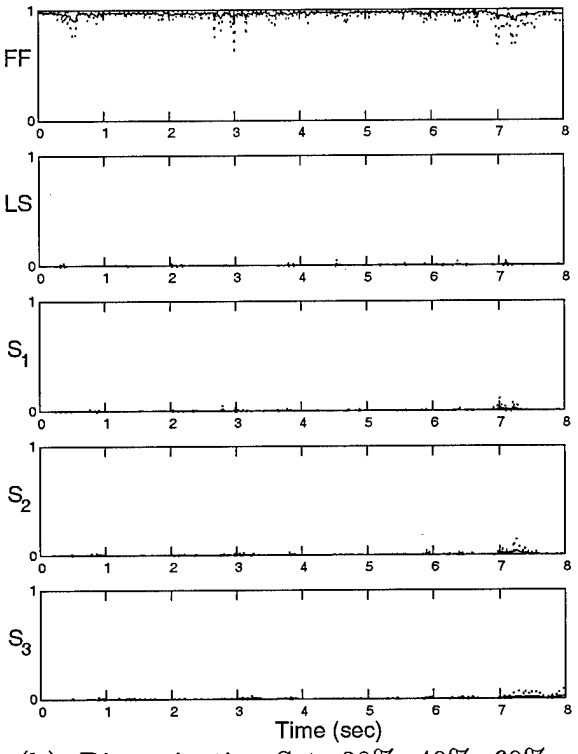


(d) Discretization Set: 40%, 60%, 80%

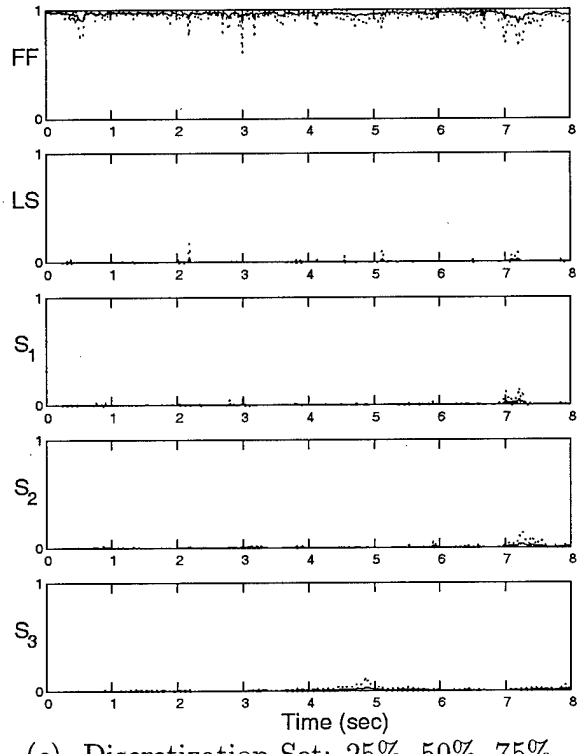
Figure 15. Probability Plot: Left Stabilator Failure, $\epsilon = 70\%$



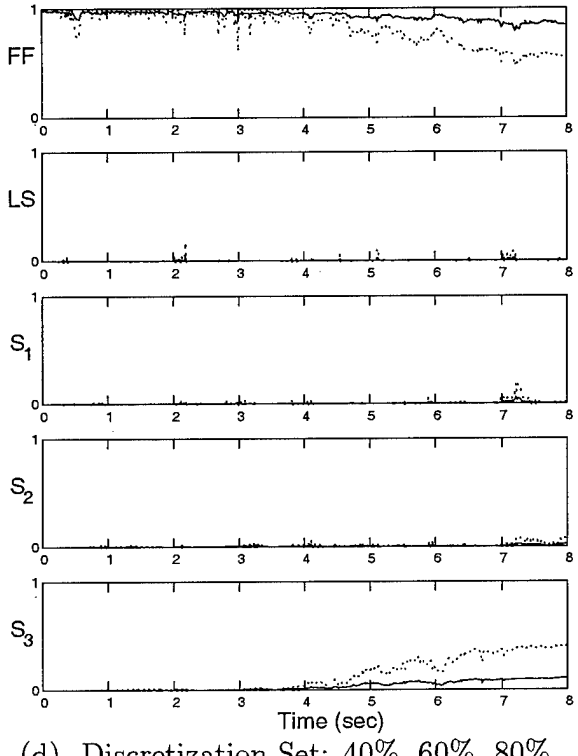
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

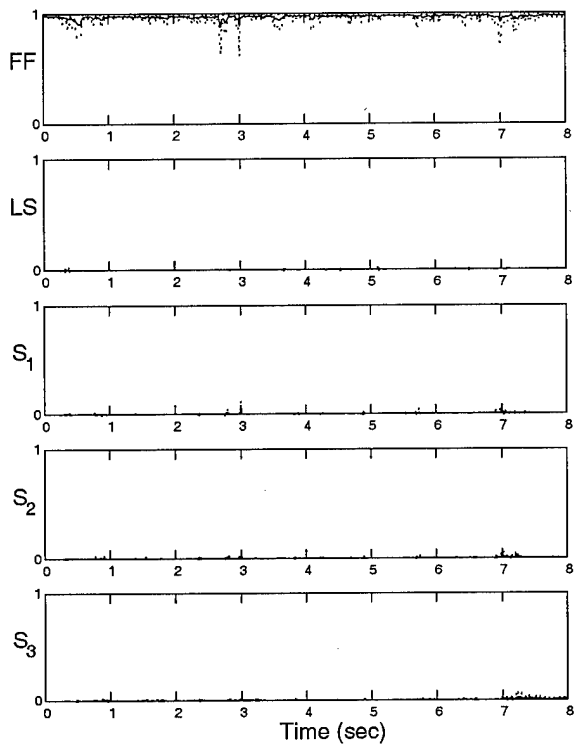


(c) Discretization Set: 25%, 50%, 75%

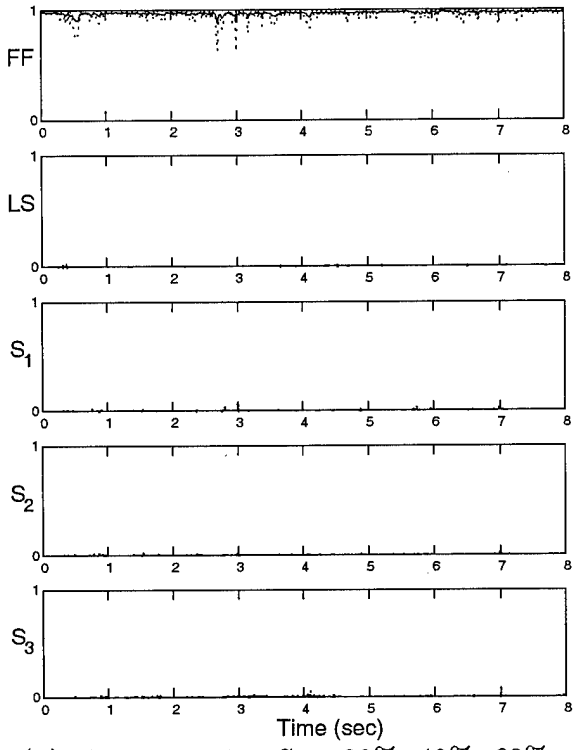


(d) Discretization Set: 40%, 60%, 80%

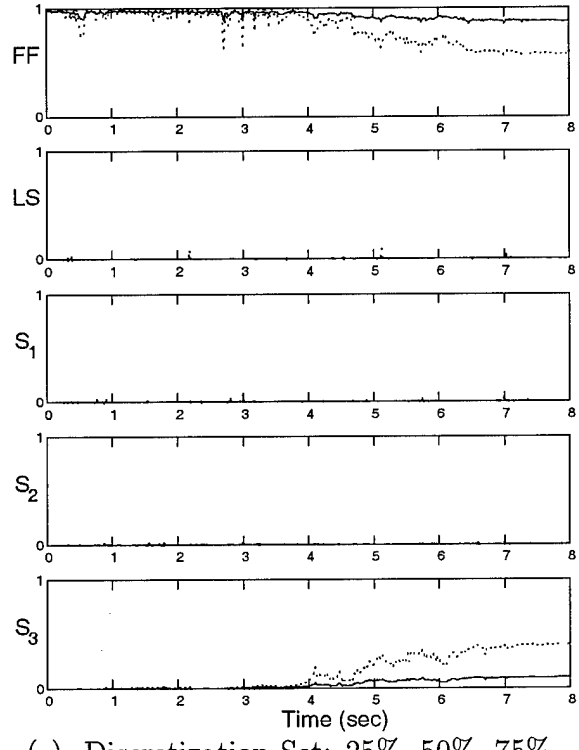
Figure 16. Probability Plot: Left Stabilator Failure, $\epsilon = 80\%$



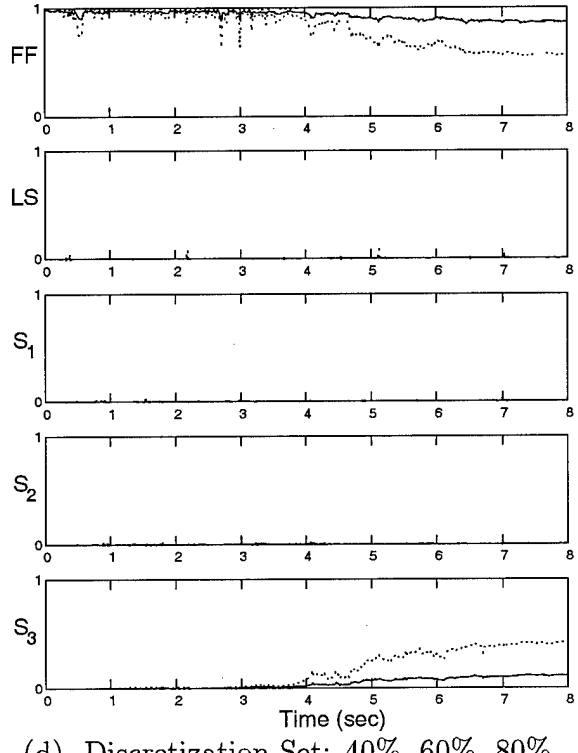
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

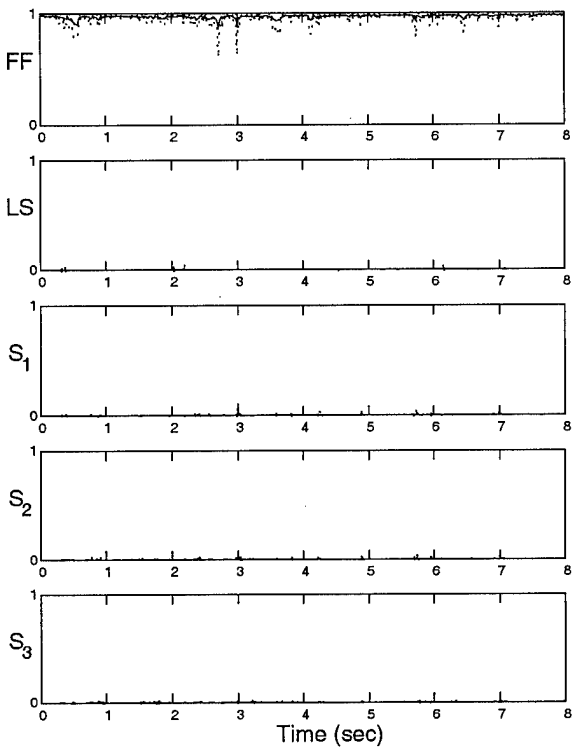


(c) Discretization Set: 25%, 50%, 75%

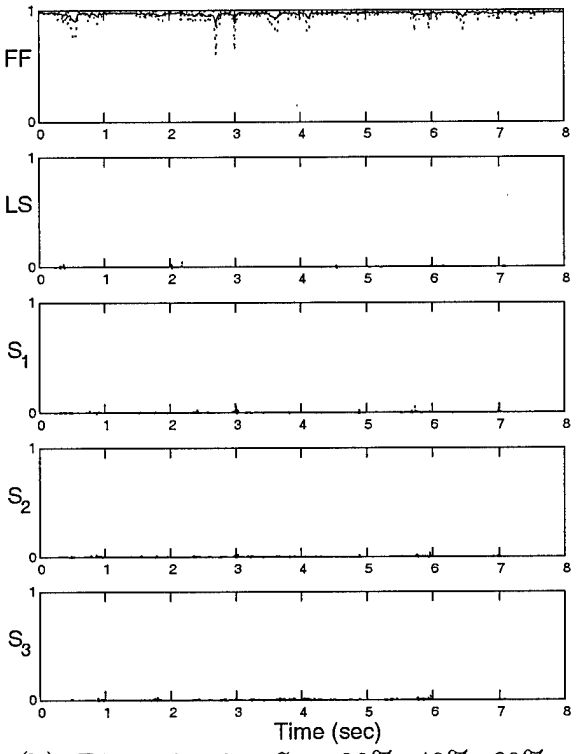


(d) Discretization Set: 40%, 60%, 80%

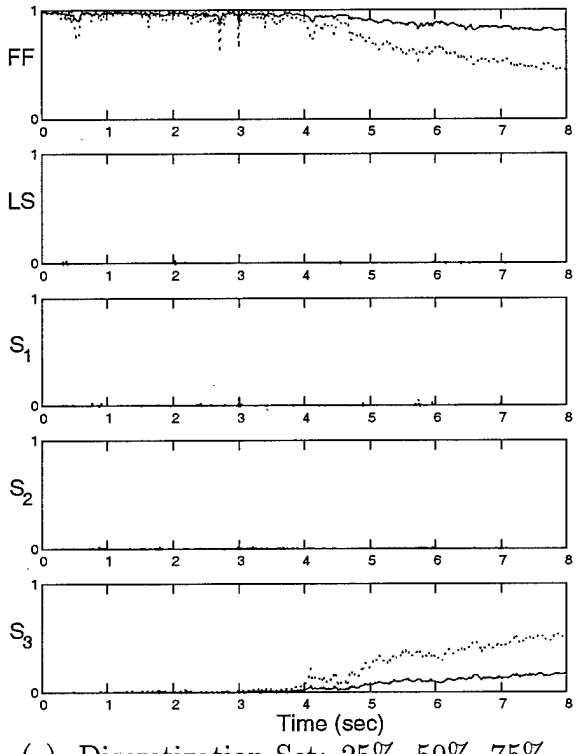
Figure 17. Probability Plot: Left Stabilator Failure, $\epsilon = 90\%$



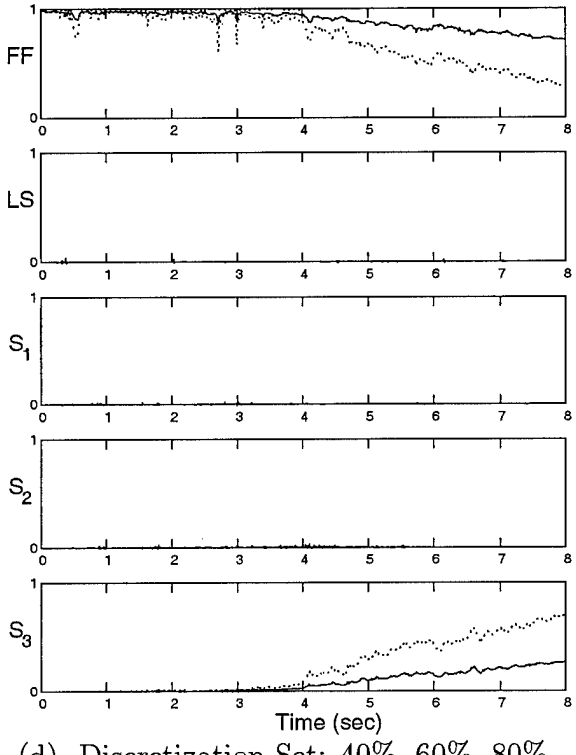
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



(d) Discretization Set: 40%, 60%, 80%

Figure 18. Probability Plot: Left Stabilator Failure, $\epsilon = 100\%$

5.3.3 *Analysis*

While much can be said about these probability plots, an analysis is provided later. The probability plots are used to clarify the plots that follow in Sections 5.4 and 5.6 and will be discussed further within those sections.

5.4 *Parameter Estimate Versus Time Plot*

5.4.1 *Description*

In Section 5.3, the probability plots for the left stabilator are shown for each effectiveness value in Equation (5.1). In Appendix B, the probability plots for the right stabilator, left flaperon, right flaperon, and rudder are shown for each effectiveness value in Equation (5.1). Recall from Equation (4.24) that the effectiveness estimate, $\hat{\epsilon}$, is formed as

$$\hat{\epsilon}(t_i) = \frac{p_j(t_i) + \epsilon_{j1} p_{13}(t_i) + \epsilon_{j2} p_{14}(t_i) + \epsilon_{j3} p_{15}(t_i)}{p_j(t_i) + p_{13}(t_i) + p_{14}(t_i) + p_{15}(t_i) + p_1(t_i)} \tag{5.2}$$

where p_1 is the conditional probability of the fully functional filter; p_j is the conditional probability of the complete actuator failure; p_{13}, p_{14}, and p_{15} are the conditional probabilities of the spawned filters; and $\epsilon_{j1}, \epsilon_{j2}$, and ϵ_{j3} are the hypothesized effectiveness values of the spawned filters. The conditional probabilities for each filter at all times t_i are given in the probability plots. (Recall the discrete time t_i is used because the probabilities are found 64 times a second, not continuously.) The effectiveness values of the spawned filters correspond to the discretization set. Thus, for each failure, Equation (5.2) is used to obtain an effectiveness estimate[22] at each time sample.

The "parameter estimate versus time plot" is a plot of the parameter estimate found by the algorithm over time, where the true effectiveness values in Equation (5.1) were considered. Ten Monte Carlo runs were performed for each simulation. The mean is plotted as solid line, and $(\text{mean} \pm 1\sigma)$ bounds are plotted as dash-dot lines. The parameter estimate generated by the MMAE with Filter Spawning algorithm at each time sample was written to a data file. The only post-processing was computing the mean and standard deviation over the 10 Monte Carlo runs.

[22]The parameter estimate consists of the failure index (i.e., identifying which actuator or sensor has failed) and the effectiveness in the case of an actuator failure. Here, since the algorithm has correctly detected the proper actuator failure, the remaining parameter estimate consists of an estimate of the effectiveness. Thus, parameter estimate and effectiveness estimate are used interchangeably.

5.4.2 *Results*

The parameter estimate mean$\pm 1\sigma$ versus time is computed for each failure. The *left stabilator* is shown here, where the figure and page numbers are listed in Table 13. The parameter estimates versus time for the remaining failures are shown in Appendix C. In each figure of this section and in Appendix C, the subplots vary the true effectiveness from 0% to 100% in steps of 10%.

Table 13. Figure Numbers for Parameter Estimates Versus Time Plots

Figure	Discretization Set	Page
19	#1 – 10%, 25%, 50%	87
20	#2 – 20%, 40%, 60%	88
21	#3 – 25%, 50%, 75%	89
22	#4 – 40%, 60%, 80%	90

5.4.3 *Analysis*

All of the information contained in parameter estimate versus time plots can be extracted from the probability plots. Moreover, the probability plots indicate in detail *how* the MMAE produces its parameter estimate through the mechanism of probability flow to the individual filters within the MMAE architecture. The parameter estimate versus time plots show the blended estimate at each time, using the conditional probabilities given in the probability plots for each hypothesized parameter value. For instance, compare Figure 12(a) to Figure 19($\mathbf{a}_{true} = 40\%$). In Figure 12(a), from 0 to 1 second, the fully functional filter (FF) contains most of the probability. Thus, the reader can expect the estimate to be around 100%, as shown in Figure 19($\mathbf{a}_{true} = 40\%$). In Figure 12(a), from 1 to 7.5 seconds, the fully functional (FF) filter slowly gives its probability to the third spawned filter (S_3), with a 50% failure hypothesis. Notice the large $\pm\sigma$ bounds indicate that the probability exchange varied considerably for each Monte Carlo run. Now turn to Figure 19($\mathbf{a}_{true} = 40\%$). From 1 to 7.5 seconds, the estimate slowly goes from 100% to 50%. The large $\pm\sigma$ bounds indicate that the estimate varied considerably for each Monte Carlo run. Finally, in Figure 12(a) from 7.5 to 8 seconds, the third spawned filter (S_3) contains most of the probability over all of the Monte Carlo runs. In Figure 19($\mathbf{a}_{true} = 40\%$), the estimate is at 50% with very small (unnoticeable at this scale) $\pm\sigma$ bounds.

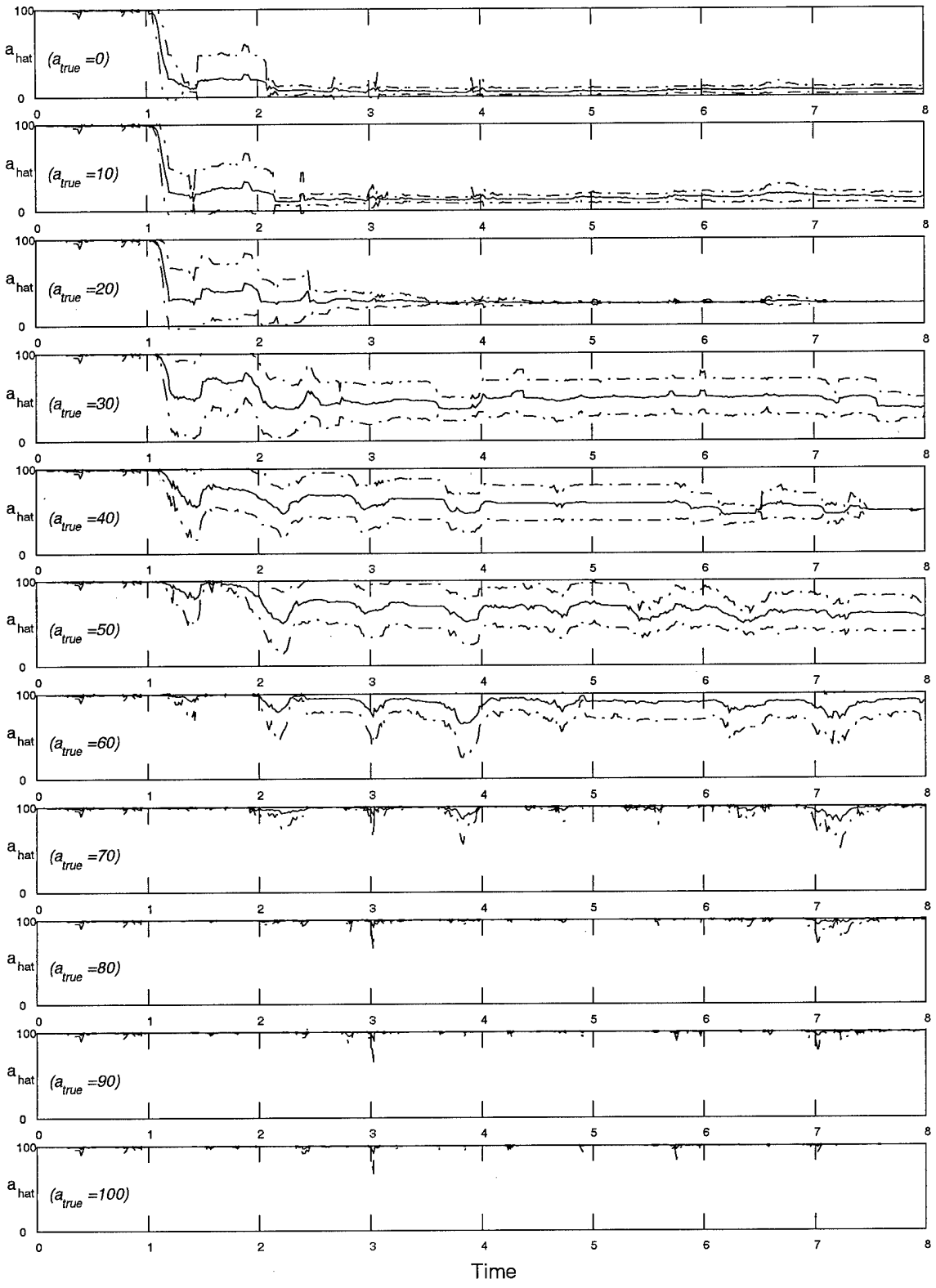


Figure 19. â Versus Time Plots for Left Stabilator Failure Using Spawned Filters 10%, 25%, 50%.

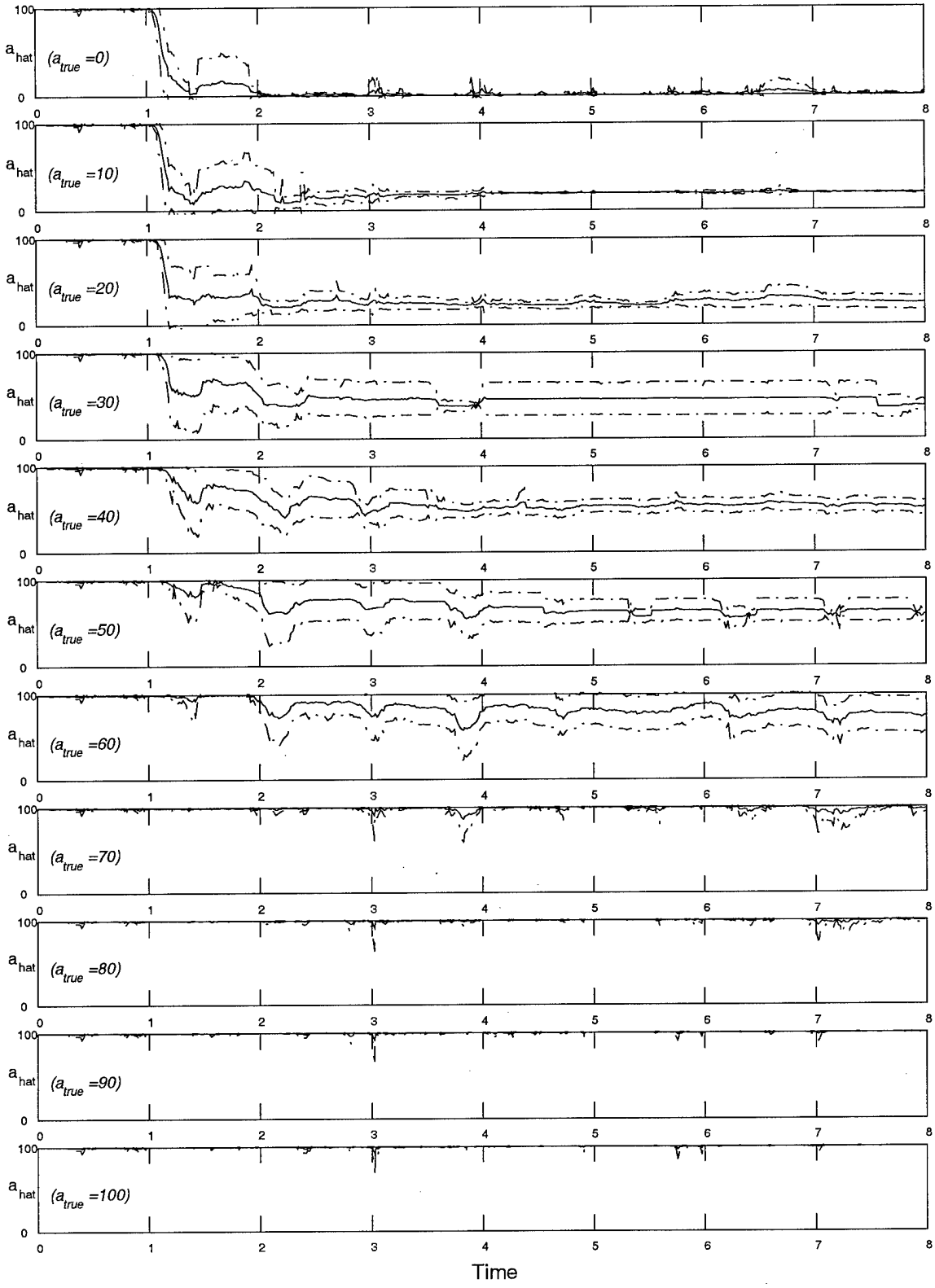


Figure 20. â Versus Time Plots for Left Stabilator Failure Using Spawned Filters 20%, 40%, 60%.

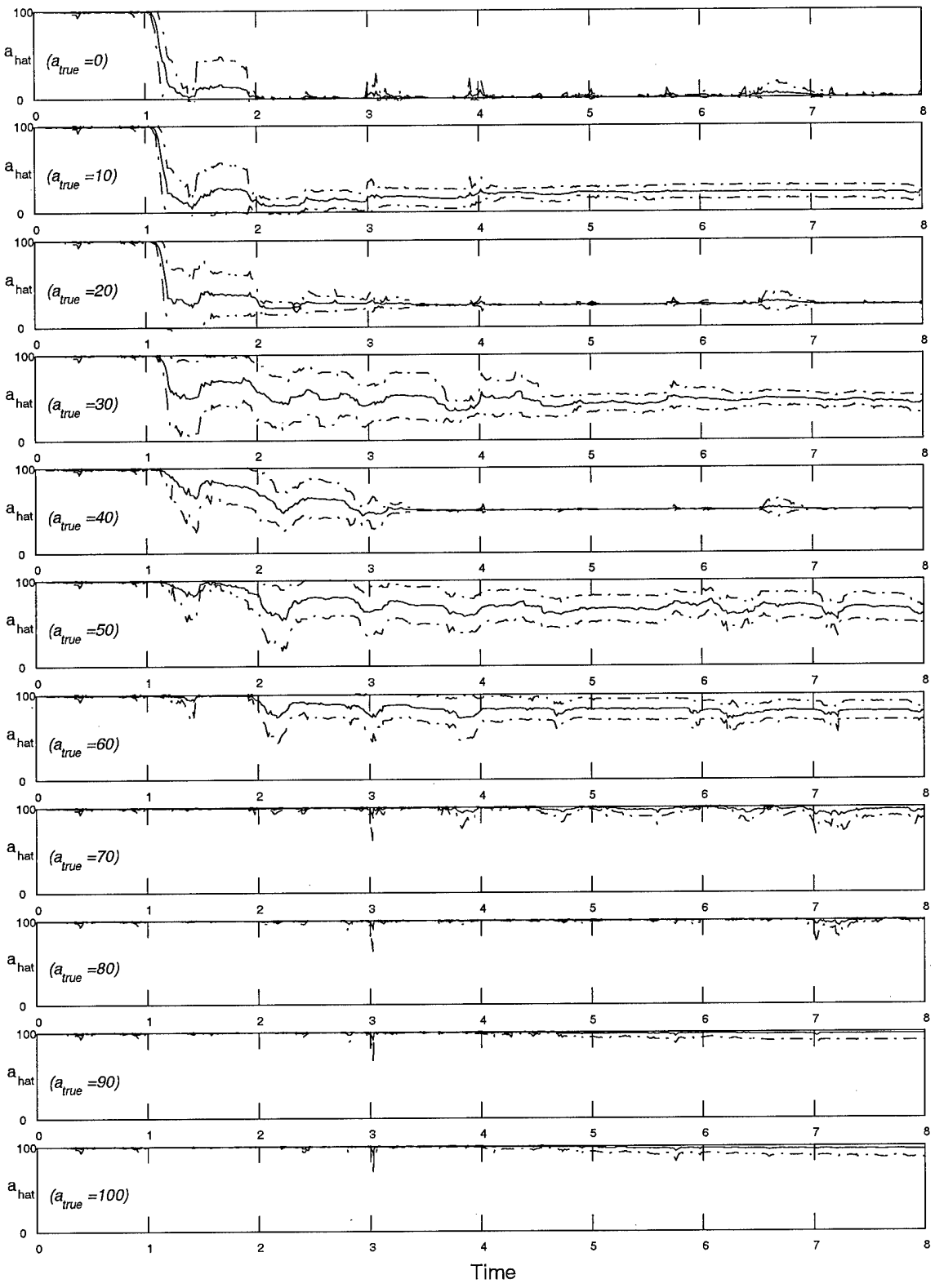


Figure 21. â Versus Time Plots for Left Stabilator Failure Using Spawned Filters 25%, 50%, 75%.

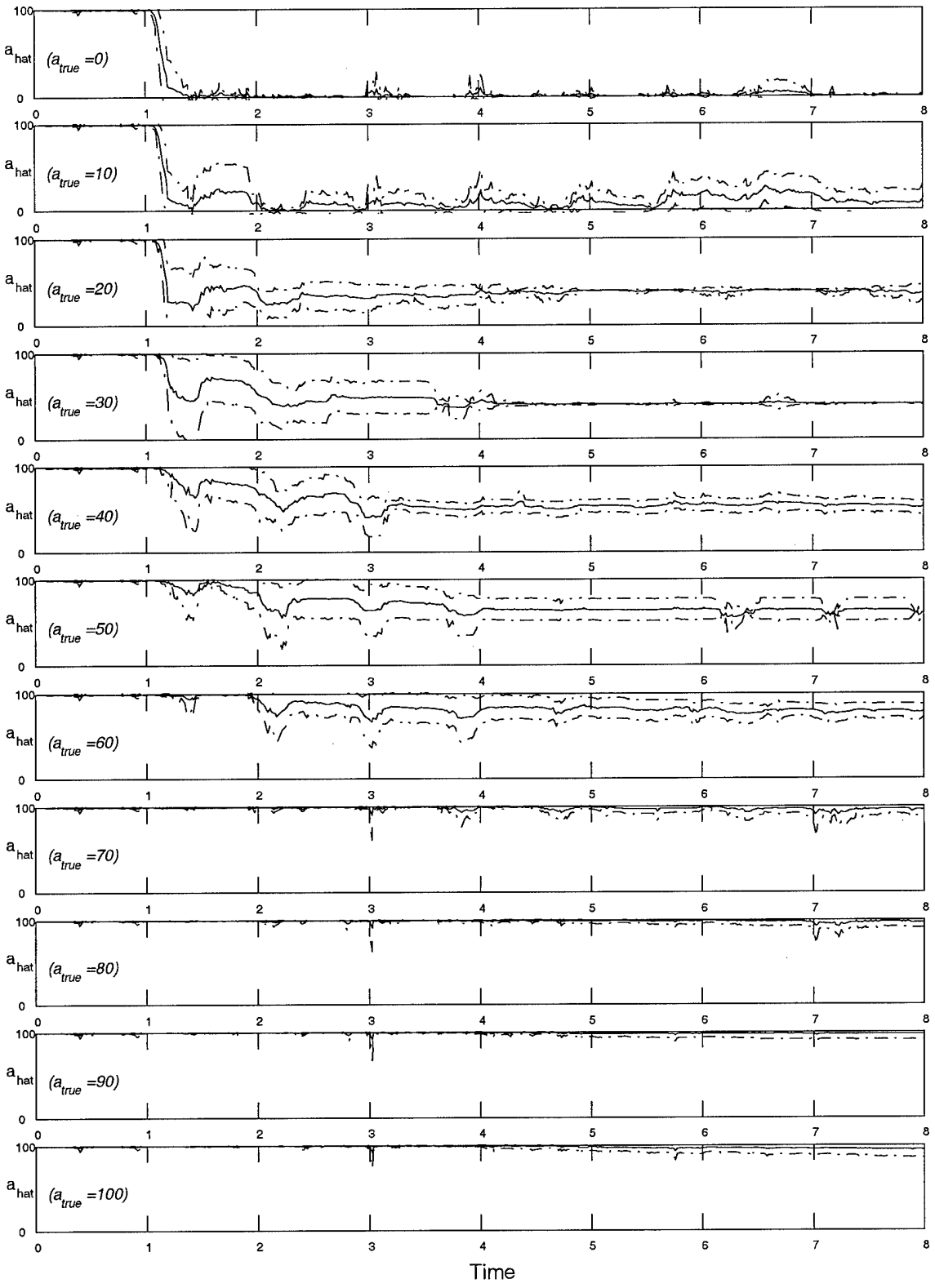


Figure 22. â Versus Time Plots for Left Stabilator Failure Using Spawned Filters 40%, 60%, 80%.

Plotting the parameter estimate versus time shows clearly the *convergence rate* of the parameter estimate[23]. The mean *and* $\pm 1\sigma$ bounds give an indication of the time to converge to a solution. Notice the estimate converges by $t_i \approx 4$ seconds for each discretization set. Recalling the failure is introduced at 1 second, the estimate converges by 3 seconds after the failure for each discretization set. This information is very useful in deciding when to use the parameter estimate. While the plots do not indicate it, the correct failure is detected in about one second. (In other words, a correct indication of *which* actuator has failed is found by $t_i \approx 2$ sec.) Thus, after a failure is *detected* (one second after the failure), the algorithm must wait about two seconds (three seconds after the failure) before an accurate *estimate* of actuator effectiveness (or its complement, degree of failure) is obtained.

5.5 Parameter Estimate Versus True Parameter Plots

5.5.1 Description

Just as the parameter estimates were found for $\epsilon_{true} = 0, 10, 20, \cdots, 100$ in Section 5.4, a parameter estimate could be found for a more complete set of true parameter estimates. A parameter estimate (mean and standard deviation) versus time was found for each true parameter value

$$\epsilon_{true} = 0, 1, 2, \cdots, 100 \tag{5.3}$$

Thus, to show a more complete analysis, a plot of the parameter estimate versus time could be shown for all 101 values of the true parameter. Rather than do so and overwhelm the reader (or designer) with many such plots, the information is instead compacted into a digestible and useful form. First, the parameter estimates' mean and standard deviations are averaged over time for each true parameter value. The time-average starts when the estimate has essentially converged. For simplicity, this value is selected as 4 seconds for all true parameter values and discretization sets. The time-average is taken over the remainder of the simulation, i.e., from 4 seconds to 8 seconds.

[23]Previous research efforts could rely on the probability plots to "see" parameter estimation convergence rates because blending was not used.

The time-averaged mean parameter estimate is expressed as

$$\hat{\mathbf{a}}\left(\mathbf{a}_{true}\right) = \frac{\sum_{t_i=4 \text{ sec}}^{8 \text{ sec}} \overline{\hat{\mathbf{a}}}\left(t_i, \mathbf{a}_{true}\right)}{(8-4)\cdot 64} \tag{5.4}$$

where there are $(8-4)\cdot 64 = 256$ data points, and where $\overline{\hat{\mathbf{a}}}\left(t_i, \mathbf{a}_{true}\right)$ is the Monte Carlo sample mean of the parameter estimate $\hat{\mathbf{a}}\left(t_i, \mathbf{a}_{true}\right)$. Notice that the time-averaged mean parameter estimate is expressed explicitly as a function of the true parameter. (The parameter estimate is also a function of the discretization set.)

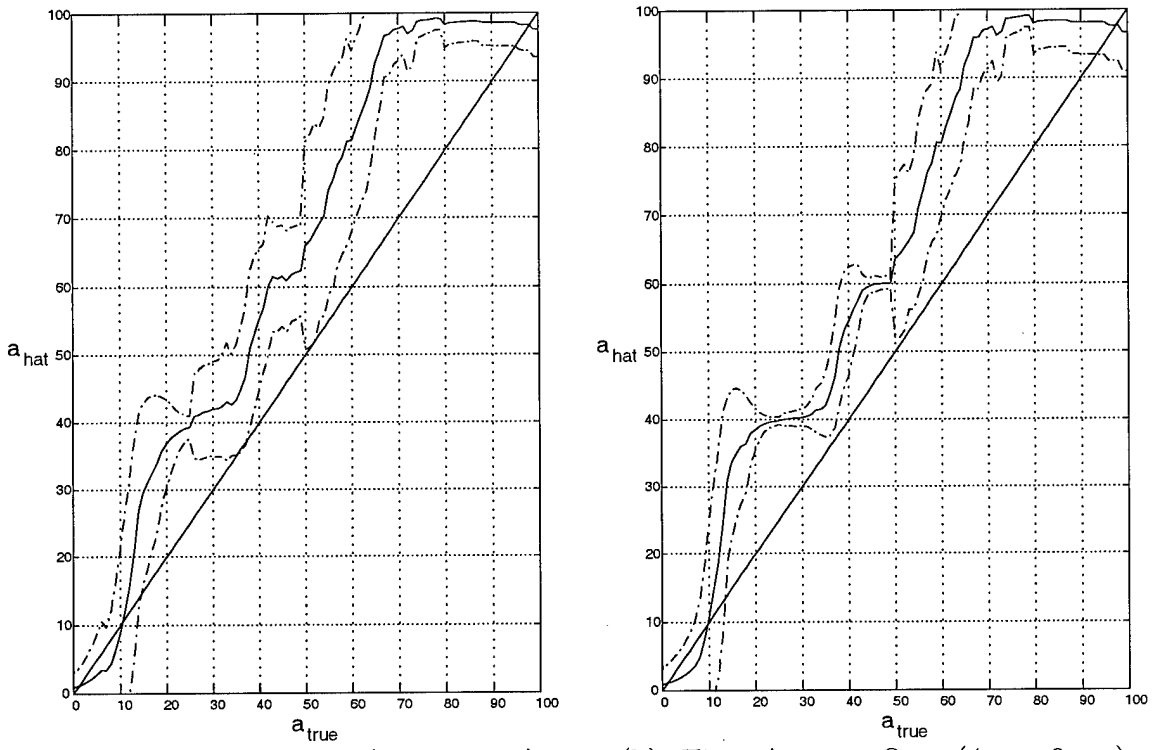
The MMAE with Filter Spawning algorithm computes the parameter estimate at each time sample and outputs it to a data file. This is done for each true effectiveness given in Equation (5.3). Post-processing the data, the mean and standard deviation of each parameter estimate at each time t_i is computed over 10 Monte Carlo runs. Finally, a time-averaged mean$\pm 1\sigma$ parameter estimate is found using Equation (5.4) for each true effectiveness given in Equation (5.3). Specifically, the *time-averaged mean* of the parameter estimate is found using Equation (5.4) where the mean (over the 10 Monte Carlo runs) of $\hat{\mathbf{a}}\left(t_i, \mathbf{a}_{true}\right)$ is used, and the *time-averaged standard deviation* of the parameter estimate is found using Equation (5.4) where the standard deviation (over the 10 Monte Carlo runs) of $\hat{\mathbf{a}}\left(t_i, \mathbf{a}_{true}\right)$ is used.. The *time-averaged mean* is plotted as a solid line, and the *time-averaged* $\pm 1\sigma$ bounds are plotted as dash-dot lines symmetrically displaced about that mean.

Ideally, the parameter estimate would equal the true parameter for all true parameter values. This plot is shown as the solid line, $\mathbf{a}_{hat} = \mathbf{a}_{true}$, to guide the reader's eye to the deviations from the ideal case.

5.5.2 Results

First, Figure 23 demonstrates the effect of starting at the time of convergence. In Figure 23(a), the time-average is naively started when the failure is detected (i.e., $t_i = 2$ sec.). In Figure 23(b), the time-average is based on the information obtained from the parameter estimate versus time plot and is started two seconds after the failure is detected (i.e., $t_i = 4$ sec.). Notice that the $\pm 1\sigma$ bounds are much smaller in Figure 23(b). The mean also shifts slightly and flattens more noticeably at the discretization levels. (The spawned filter discretization levels were 40%, 60%, and 80%, as

in Figure 22.) Thus, the information gained by plotting the parameter estimate versus time (i.e., the time to reach essential convergence) reduces the uncertainty in the time-averaged parameter estimate.



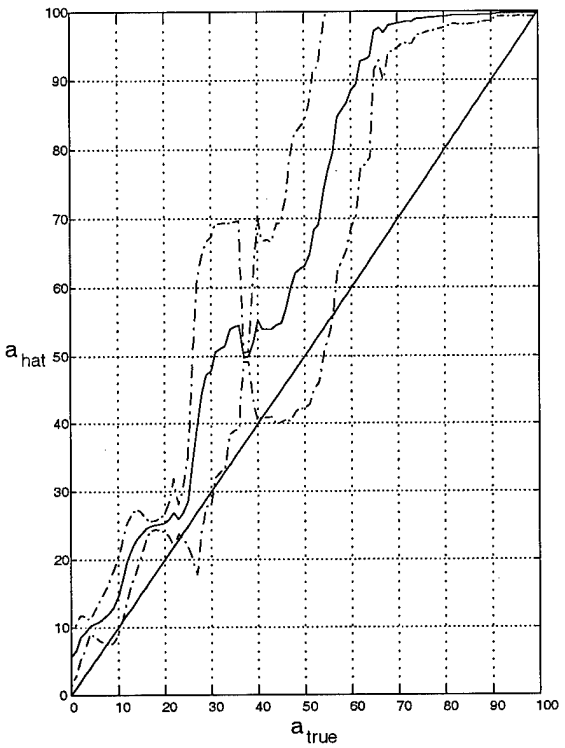
(a) Time-Average Over (2 sec, 8 sec) (b) Time-Average Over (4 sec, 8 sec)

Figure 23. Estimated Versus True Paramter Plot: Comparision of Time-Average Range

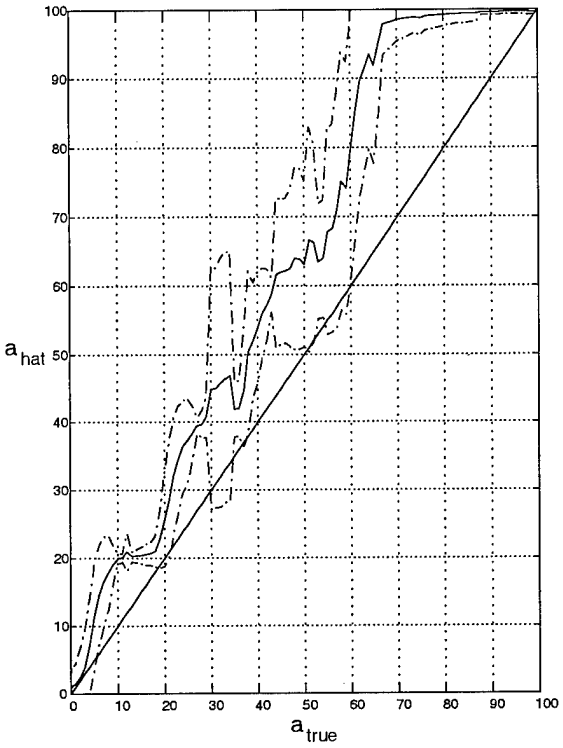
Second, the estimated versus true parameters are shown for each failure and discretization set. Figures 24 through 28 show the left stabilator failure, right stabilator failure, left flaperon failure, right flaperon failure, and rudder failure, respectively. The subplots (a) through (d) of each figure show the parameter estimate versus the true parameter using each discretization set defined in Table 11.

5.5.3 Analysis

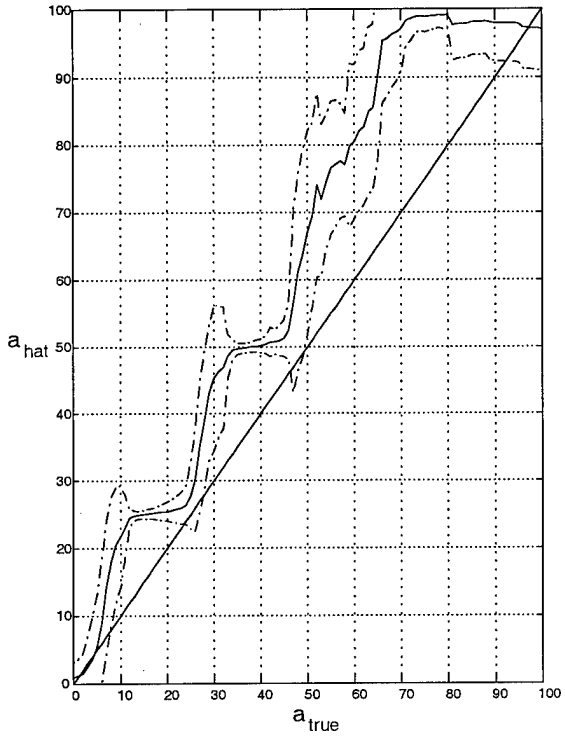
The information contained in the "probability plots" and the "parameter estimate versus time plots" is presented in five "parameter estimate versus true parameter plots" for each discretization set. Because of the compact representation, conclusions can be drawn easily with one plot instead of



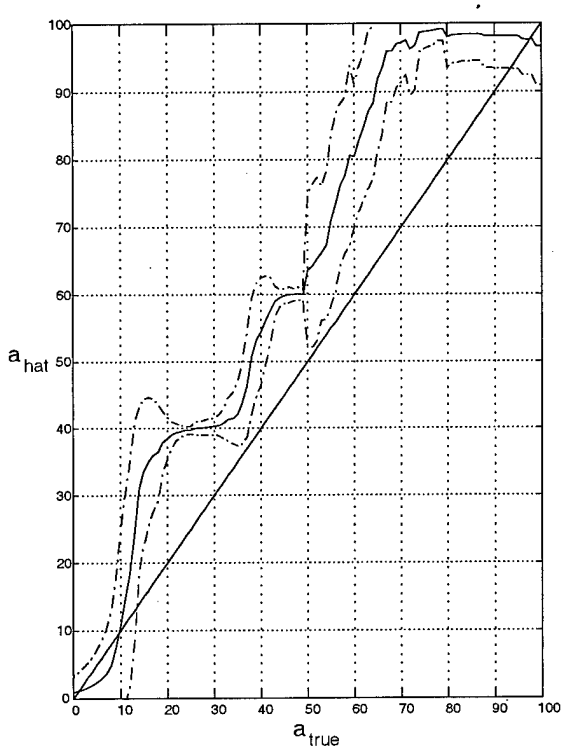
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

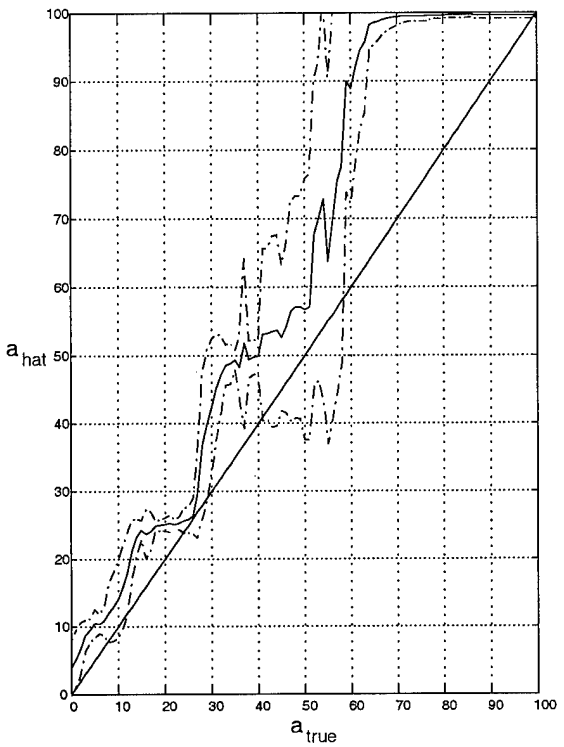


(c) Discretization Set: 25%, 50%, 75%

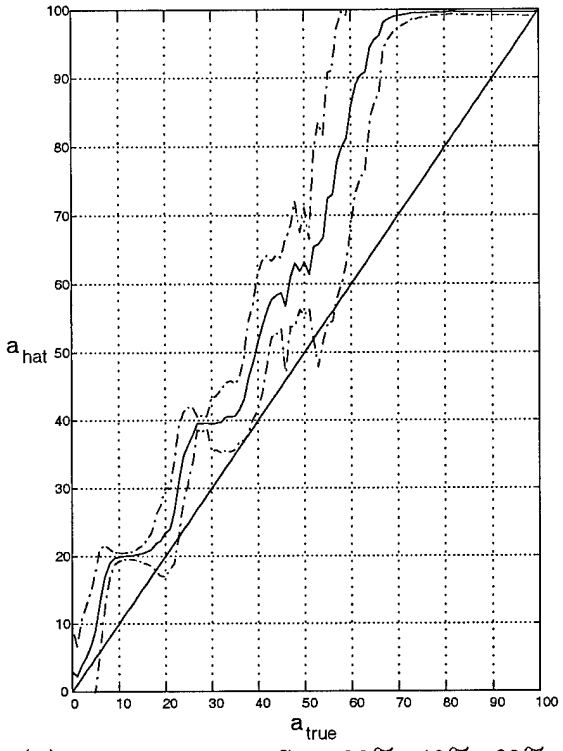


(d) Discretization Set: 40%, 60%, 80%

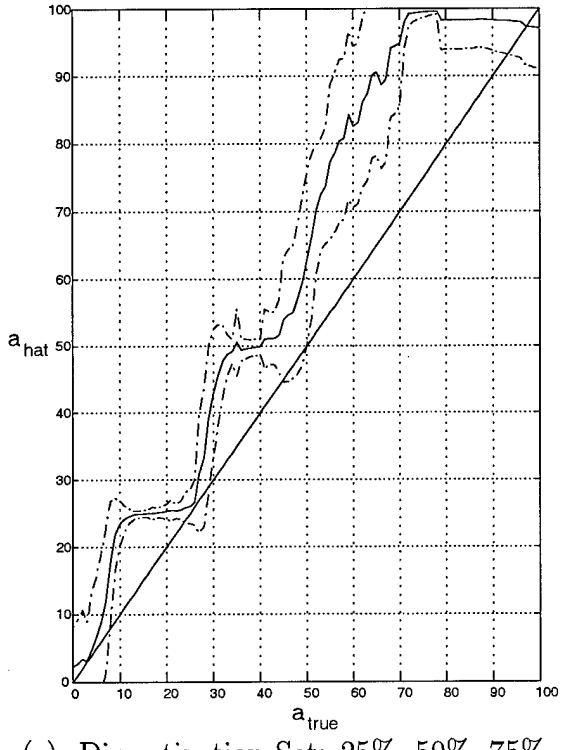
Figure 24. Estimated Versus True Parameter: Left Stabilator Failure



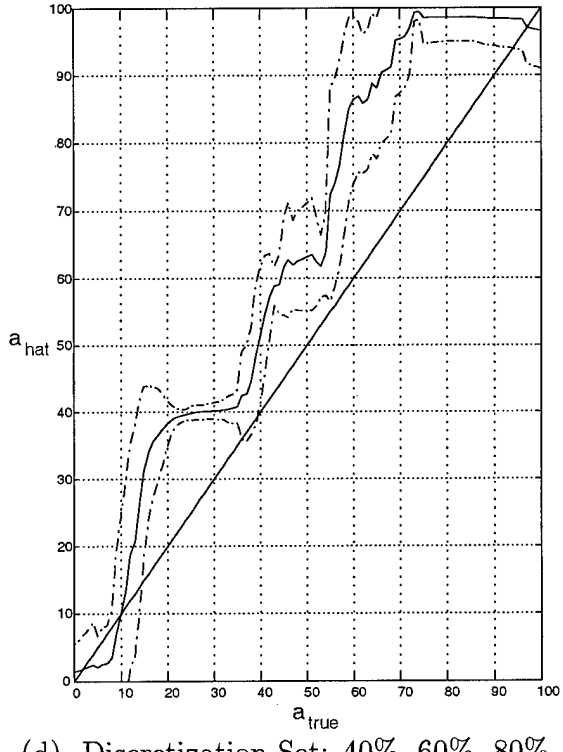
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

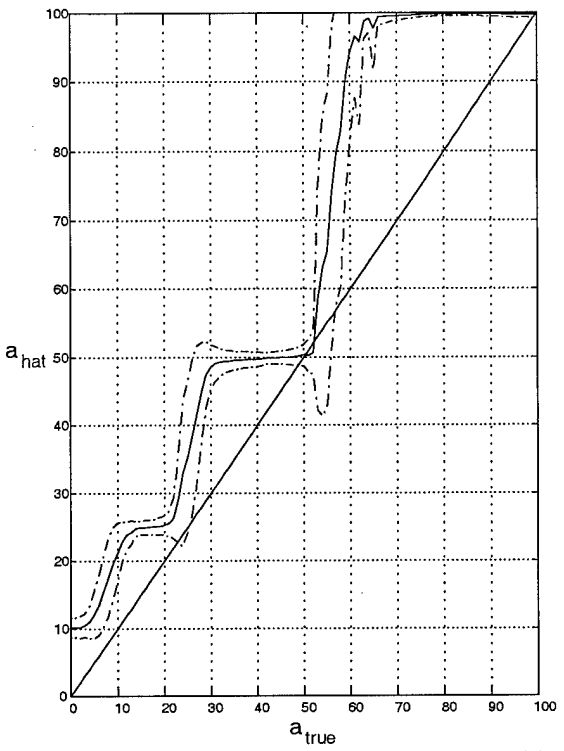


(c) Discretization Set: 25%, 50%, 75%

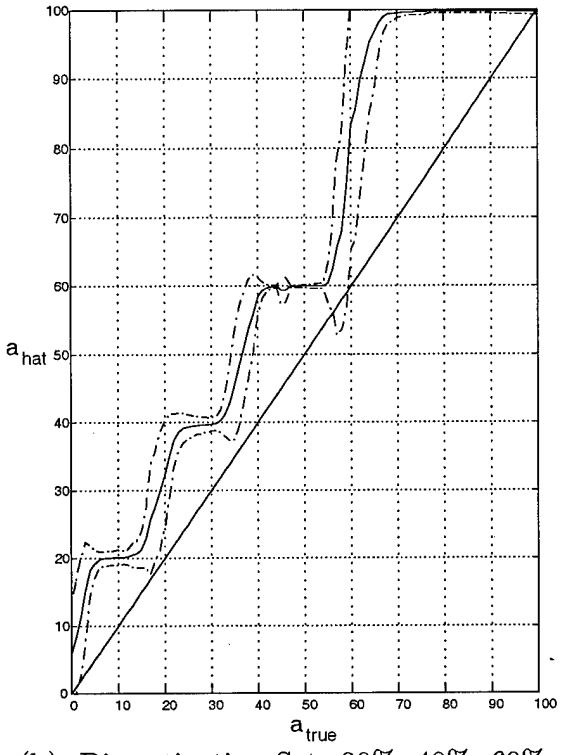


(d) Discretization Set: 40%, 60%, 80%

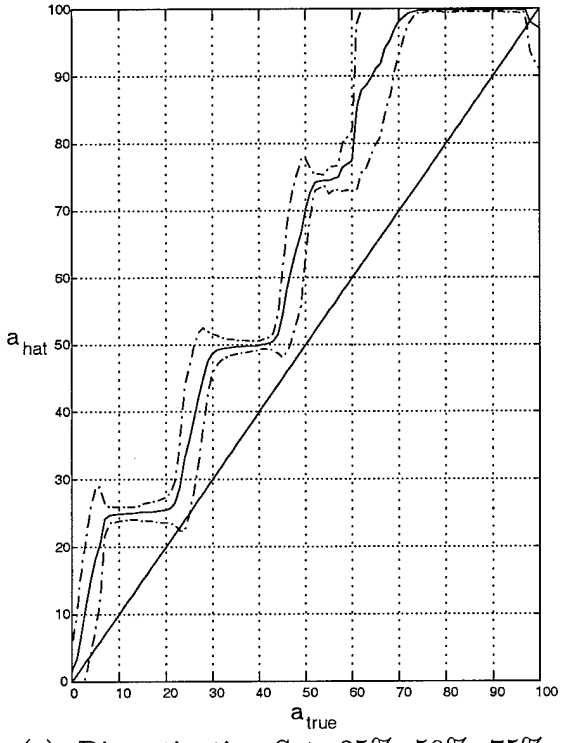
Figure 25. Estimated Versus True Parameter: Right Stabilator Failure



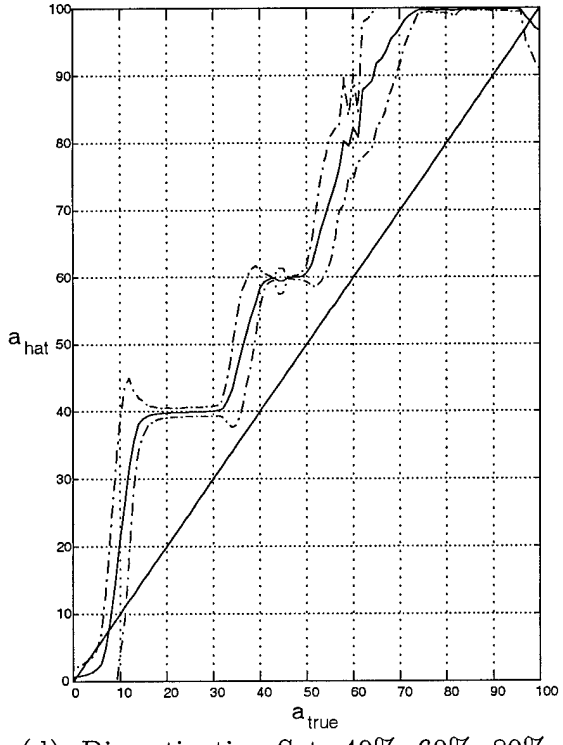
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

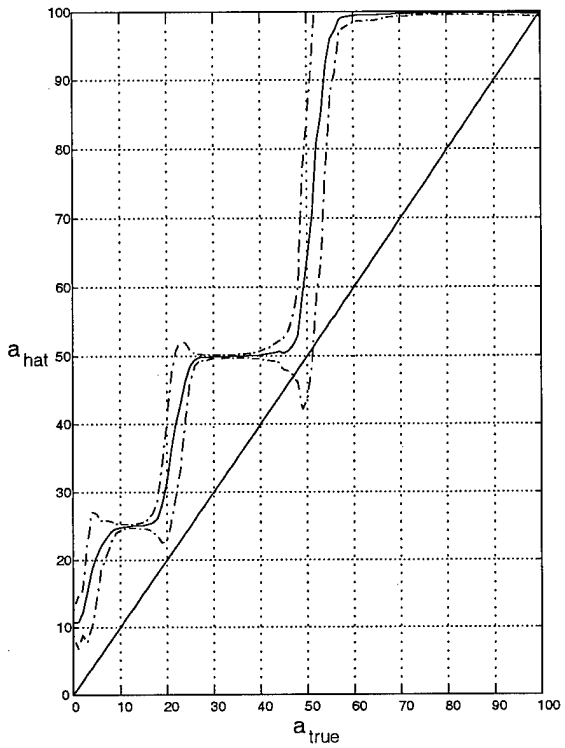


(c) Discretization Set: 25%, 50%, 75%

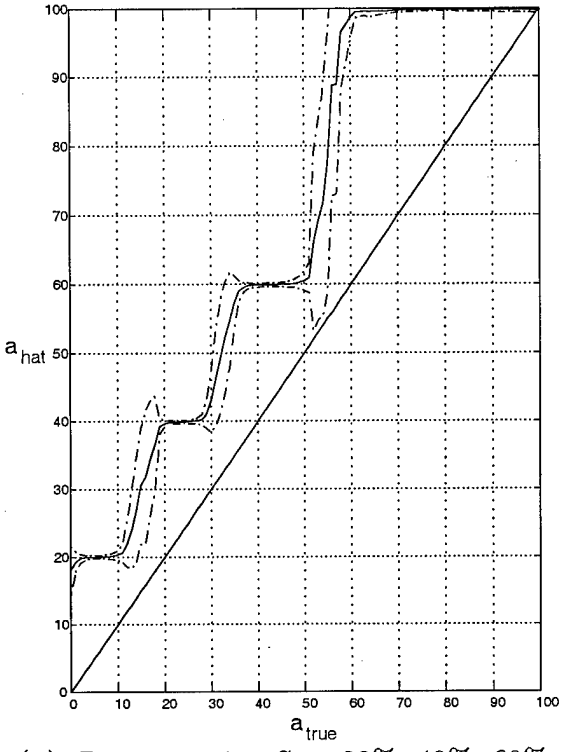


(d) Discretization Set: 40%, 60%, 80%

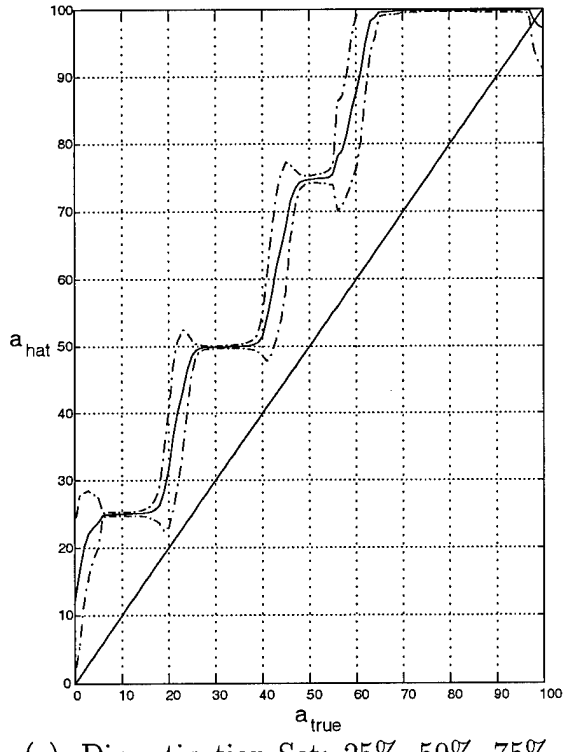
Figure 26. Estimated Versus True Parameter: Left Flaperon Failure



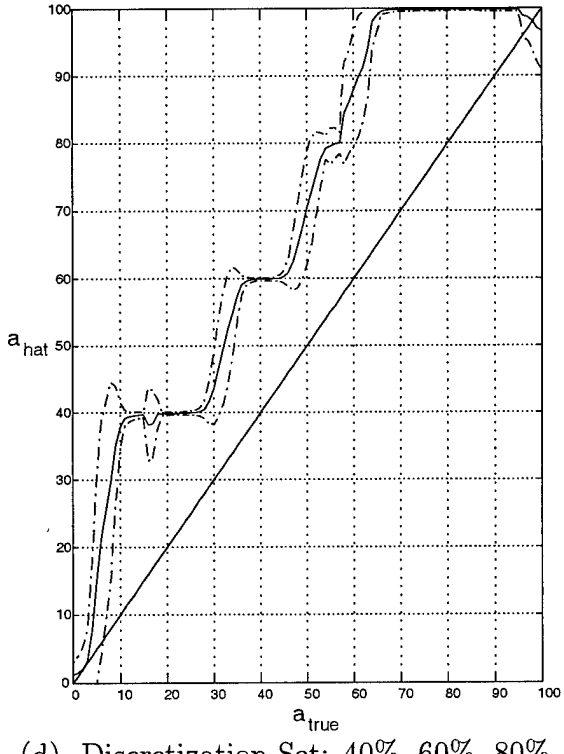
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

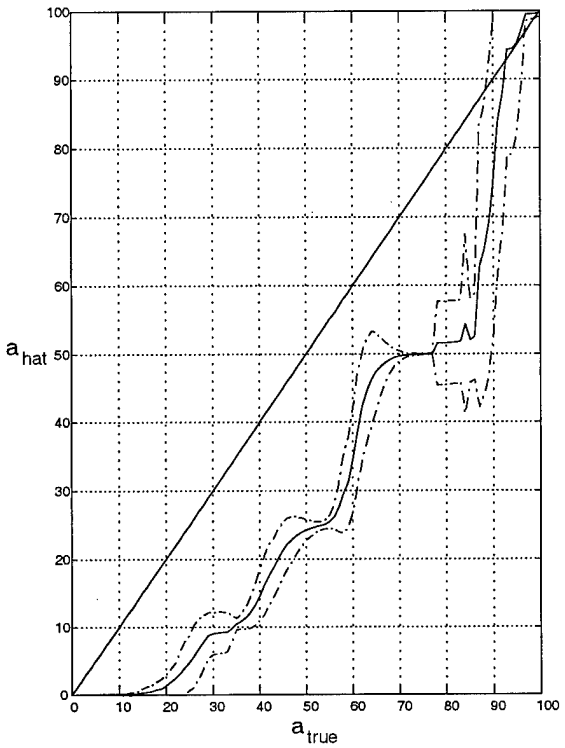


(c) Discretization Set: 25%, 50%, 75%

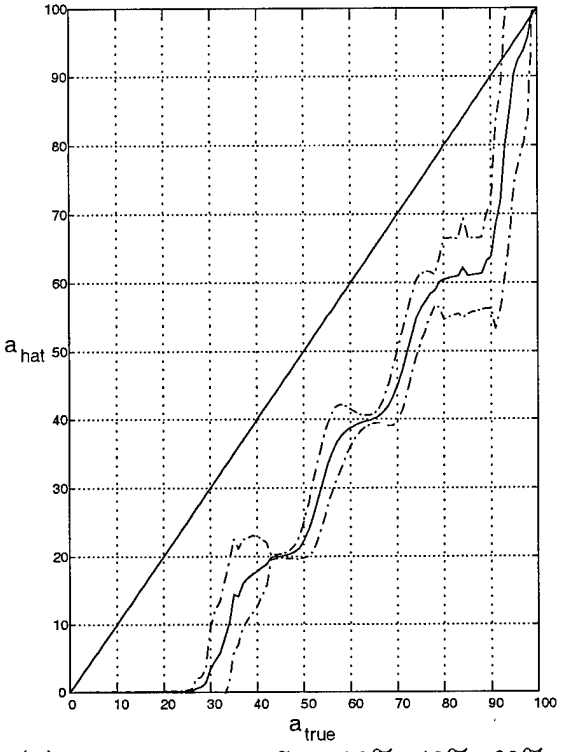


(d) Discretization Set: 40%, 60%, 80%

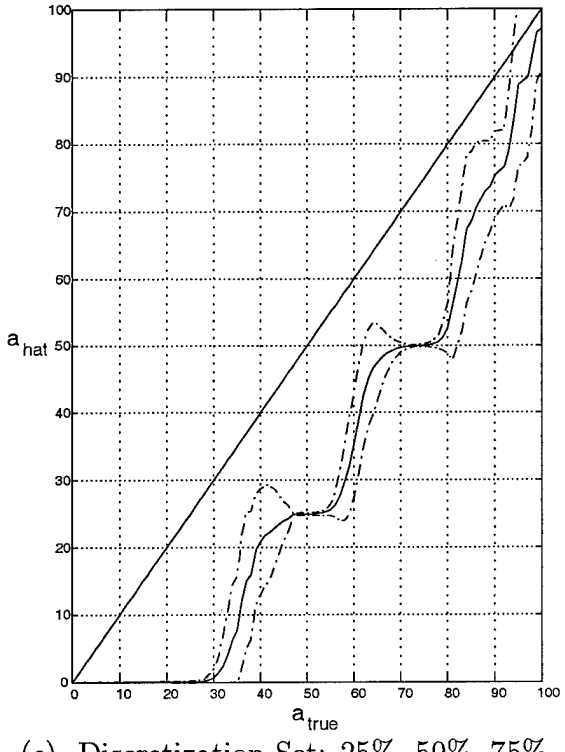
Figure 27. Estimated Versus True Parameter: Right Flaperon Failure



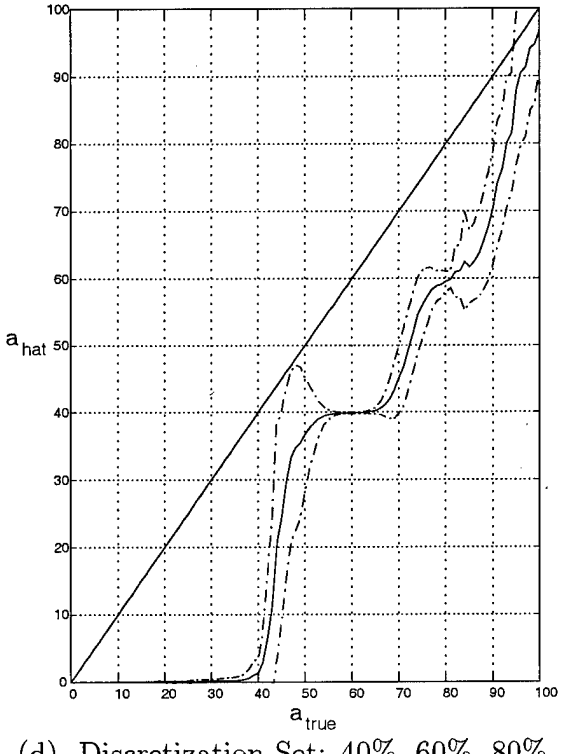
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



(d) Discretization Set: 40%, 60%, 80%

Figure 28. Estimated Versus True Parameter: Rudder Failure

comparing multiple plots. The following conclusions are made (note that some references compare multiple plots so the reader can better understand the information contained in the "parameter estimate versus true parameter plot"):

1. *Convergence to Complete Failure Hypothesis:* Because Control Redistribution uses a different computational philosophy for complete versus partial failures (see Section 3.6), it is important that for $\mathbf{a}_{true} = 0\%$, the parameter estimate indicates a complete actuator failure. Figure 24 shows that, in going from a spacing $0\%, 10\%$ in (a) to a spacing $0\%, 25\%$ in (b), the algorithm converges to the complete actuator failure estimate at $\mathbf{a}_{true} = 0\%$. This is also seen in Figure 8(a), noticing that the first spawned filter (S_1) at 10% shares the probability with the complete left stabilator failure filter (LS), but in Figure 8(b), the first spawned filter (S_1) at 25% *does not* share the probability with the complete left stabilator failure filter (LS). Likewise, in Figure 19($\mathbf{a}_{true} = 0$) the estimate is *not* zero, but in Figure 20($\mathbf{a}_{true} = 0$) the estimate is zero. (Notice in Figure 27 for the right flaperon, this convergence to the complete actuator failure at $\mathbf{a}_{true} = 0\%$ does not occur until the separation $(0\%, 40\%)$ between the two lowest assumed $\mathbf{a}$ values is achieved. This attribute can be found in the probability plots and parameter estimate versus time plots in Appendix B as well.)

2. *Blending and Uncertainty Trade-off:* Notice in Figure 24(b) that the mean is blended well, but in Figure 24(d) the mean flattens at each discretization level (except at 80%). Although a blended estimate is more desirable, the flattened estimate's $\pm 1\sigma$ bounds are smaller. Thus, a trade-off exists between blending and uncertainty in the estimate. Consider the region 20% to 30% to see this trade-off in the probability plots. In Figures 10(d) and 11(d) the first spawned filter (S_1) at 40% consumes most of the probability with small deviations after $t_i = 4$ sec. But in Figure 10(b), the probability is shared among the first spawned filter (S_1) at 20% and the second spawned filter (S_2) at 40% with large $\pm 1\sigma$ bounds, and in Figure 11(b), the probability is shared among the second spawned filter (S_2) at 40% and the fully functional filter (FF) at 100% with large $\pm 1\sigma$ bounds. This trade-off can also be seen in the parameter estimate versus

time plots. In Figure 22, the estimates for $\mathbf{a}_{true} = 20\%$ and $\mathbf{a}_{true} = 30\%$ remain the same and the $\pm 1\sigma$ bounds are small in both cases, but in Figure 20, the estimates for $\mathbf{a}_{true} = 20\%$ and $\mathbf{a}_{true} = 30\%$ vary in proportion to $\mathbf{a}_{true}$ and the $\pm 1\sigma$ bounds are larger than Set #4's $\pm 1\sigma$ bounds in Figure 22.

3. *Poor Estimation Beyond* $\mathbf{a}_{true} = 70\%$*:* In Figures 24 through 27, no discretization set performs well for a true effectiveness greater than 70%, for all actuators except the rudder. Thus, an effectiveness greater than 70% appears as a fully functional aircraft to the algorithm, or a failure goes unnoticed until the failure's effectiveness is less than 70%. Because this attribute is undesirable, improving performance for a true effectiveness greater than 70% was attempted by using discretization sets $40\%, 60\%, 90\%$ and $70\%, 80\%, 90\%$. The result was degraded performance in the region less than 70%, with no improvement in the region beyond 70%. This effect is also seen in the probability plots for the left stabilator in Figures 15 through 18. Notice for all discretization sets (a-d), the fully functional filter (FF) receives most of the probability. The probability plots for the right stabilator, left flaperon and right flaperon in Appendix B show similar results. In addition, this effect is seen also in the parameter estimate versus time plots for the left stabilator in Figures 19 through 22 in the subplots of $\mathbf{a}_{true} = 70\%$ through $\mathbf{a}_{true} = 100\%$. The parameter estimate versus time plots for the right stabilator, left flaperon and right flaperon in Appendix C show similar results.

4. *Poor Estimation for Small Effectiveness, Rudder Failures:* In Figure 28, no discretization set performs well for a true effectiveness less than the lowest hypothesized effectiveness value for the rudder. Thus, small values of true effectiveness for a rudder failure appear to the algorithm as a complete rudder failure. When the estimate is used in Control Redistribution, it is acceptable to have small effectiveness values appear as complete rudder failures. This does not allow the rudder to be used when some control authority could still be obtained; however, the designer must admit that using an actuator with a small effectiveness value may only degrade the actuator further. This effect is seen in the probability plots in Appendix B in Figures 80

through 83. For all discretization sets (a-d), the complete rudder failure filter (R) receives most of the probability. This effect is seen also in the parameter estimate versus time plots in Appendix C in Figures 96 through 99 in the subplots of $\mathbf{a}_{true} = 70\%$ through $\mathbf{a}_{true} = 100\%$.

5. *Bias in the Estimate:* Through the aid of the $\mathbf{a}_{hat} = \mathbf{a}_{true}$ line, in Figures 24 through 27 (i.e., for stabilators and flaperons) for all discretization sets (a-d), the effectiveness estimate is *larger* than the true effectiveness (except the fully functional and complete failure cases). In contrast, in Figure 28 (i.e., for the rudder) for all discretization sets (a-d), the effectiveness estimate is *smaller* than the true effectiveness (except the fully functional and complete failure cases). This effect is seen in the probability plots for the left stabilator in Figures 9 through 17. Notice for all discretization sets (a-d), a filter with a *larger* hypothesized effectiveness than the true effectiveness consumes most of the probability. The probability plots for the right stabilator, left flaperon and right flaperon in Appendix B show similar results. This effect is seen also in the parameter estimate versus time plots for the left stabilator in Figures 19 through 22 in the subplots of $\mathbf{a}_{true} = 10\%$ through $\mathbf{a}_{true} = 90\%$. The parameter estimate versus time plots for the right stabilator, left flaperon and right flaperon in Appendix C show similar results.

6. *Flaperon Estimation Better Than Stabilator Estimation:* Comparing the $\pm 1\sigma$ bounds for the flaperons in Figures 26 and 27 to the $\pm 1\sigma$ bounds for the stabilators in Figures 24 and 25, respectively, the performance is better for all discretization sets (a-d). As anticipated, this effect is also seen in the probability plots. The left stabilator in Figures 8 through 18 and the right stabilator in Figures 40 through 50 in Appendix B have larger $\pm 1\sigma$ bounds than the left flaperon in Figures 51 through 61 in Appendix B and the right flaperon in Figures 62 through 72 in Appendix B, respectively. (Notice how time-consuming it is to "see" this result looking at the numerous probability plots versus Figures 24 through 27!) Furthermore, this effect is seen also in the parameter estimate versus time plots. The left stabilator in Figures 19 through 22 in all subplots corresponding to different values of $\mathbf{a}_{true}$, and the right stabilator Figures 84 through 87 in all subplots of $\mathbf{a}_{true}$ in Appendix C, have larger $\pm 1\sigma$ bounds than

the left flaperon in Figures 88 through 91 in all subplots of $\mathbf{a}_{true}$ in Appendix C and the right flaperon in Figures 92 through 95 in all subplots of $\mathbf{a}_{true}$ in Appendix C, respectively. Notice again how time-consuming it is to "see" this result looking at the numerous plots. For a designer, plots of the form of Figures 24 through 27 are extremely useful for making design decisions.

7. *Right Channel Estimation Better Than Left Channel Estimation:* Comparing the $\pm 1\sigma$ bounds for the right channel in Figures 25 and 27 to the $\pm 1\sigma$ bounds for the left channel in Figures 24 and 26, respectively, the right channel performance is superior for all discretization sets (a-d). The improvement for the discretization sets in (a) and (b) is seen somewhat more readily because the $\pm 1\sigma$ bounds are larger than the discretization sets in (c) and (d). This effect is attributed to how the dither is applied; the phasing is different for the right and left channels, for both stabilators and flaperons. Prior research has found similar results [8,18,41,42]. As before, this effect is also seen in the probability plots. The left stabilator in Figures 8 through 18 and the left flaperon in Figures 51 through 61 in Appendix B have larger $\pm 1\sigma$ bounds than the right stabilator in Figures 40 through 50 in Appendix B and the right flaperon in Figures 62 through 72 in Appendix B, respectively. (Again, it is time-consuming to "see" this result looking at the numerous probability plots.) Also, as found previously, this effect is evident in the parameter estimate versus time plots. The left stabilator in Figures 19 through 22 in all subplots of $\mathbf{a}_{true}$ and the left flaperon in Figures 88 through 91 in all subplots of $\mathbf{a}_{true}$ in Appendix C have larger $\pm 1\sigma$ bounds than the right stabilator Figures 84 through 87 in all subplots of $\mathbf{a}_{true}$ in Appendix C and the right flaperon in Figures 92 through 95 in all subplots of $\mathbf{a}_{true}$ in Appendix C, respectively. (Again, a designer can see these trends more efficiently from the digested form of Figures 24 through 27 than from the numerous time plots.)

8. *Rudder Behavior "Opposite" Other Actuators:* Notice the rudder in Figure 28 behaves in a manner *opposite* that of the other actuators. First, for the rudder, an accurate estimate is difficult to obtain when the true parameter is near a complete failure, whereas for the other

actuators, an accurate estimate is difficult to obtain when the true parameter is near a fully functional aircraft. Second, for the rudder, the estimated parameter is consistently *smaller* than the true parameter value, whereas for the other actuators, the estimated parameter is consistently *larger* than the true parameter value. Previous research also found the rudder behaved differently from the other actuators [8]. The exact *behavioral* difference was not known until the MMAE with Filter Spawning parameter estimate versus true parameter plot was obtained. The reason for the behavioral difference can be attributed, in part, to the fact that the rudder is used very little during this flight profile and that its control authority is less than that of the other actuators.

5.5.4 Selecting A Discretization Set

The parameter estimate versus true parameter plots allow a convenient comparison among the discretization sets. To decide which discretization set is "best", the application must be taken into account. For this research, a parameter estimate is sought in order to apply Control Redistribution (CR) to maintain stability and control of the VISTA F-16 under partial and complete actuator failures. While merely *detecting* a failure is sufficient to provide limited control through complete failure CR, this research seeks to *estimate* the degree of the failure so that partial failure CR (or, modified CR) can be applied to the aircraft (hopefully resulting in a smaller degradation in performance from the fully functional aircraft than that of CR with complete failure assumptions).

With this in mind, reconsider the discretization sets. The discretization Sets #3 and #4 are favored over Sets #1 and #2 because they exhibit smaller $\pm 1\sigma$ bounds for most of the $\mathbf{a}_{true}$ region. To choose between Sets #3 and #4, compare Figure 24(c) to Figure 24(d). If an actuator's effectiveness is small, even if a good estimate of effectiveness could be obtained, the designer may choose *not* to send control authority to that actuator. As a result, discretization Set #4 shown in Figure 24(d) is chosen because it drops off sharply in the true effectiveness interval of $10\% - 20\%$. If MMAE with Filter Spawning was implemented with a different discretization set for each actuator (i.e., $\epsilon_{i_a 1}, \epsilon_{i_a 2}$, and $\epsilon_{i_a 3}$ are different for each i_a), then Set #3 would be chosen for the rudder $i_a = 5$.

In Figure 28, Set#3 (c) provides a more even estimate than Set#4 (d), closer to the $\mathbf{a}_{hat} = \mathbf{a}_{true}$ line in the $\mathbf{a}_{true}$ range of $30\% - 40\%$ and without as large a standard deviation in the region of $\mathbf{a}_{true} = 40\% - 50\%$.

5.6 Refined Parameter Estimate Plots

5.6.1 Description

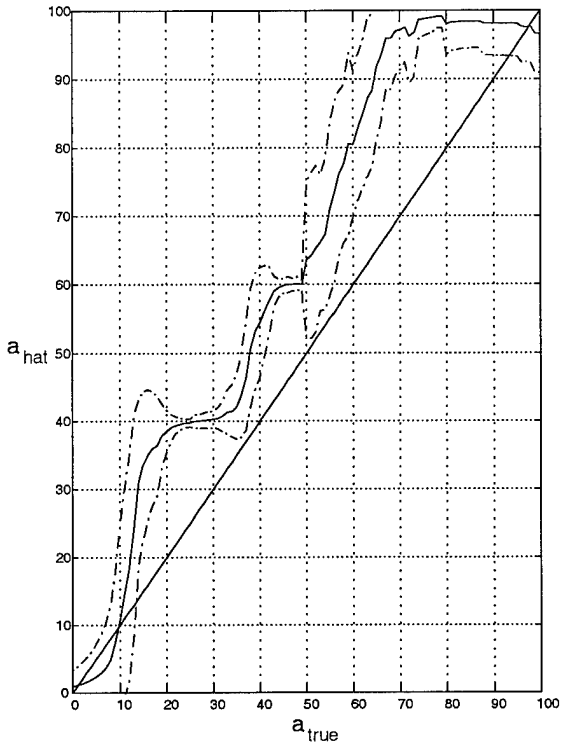
With a discretization set chosen for the spawned filters, the estimate is *refined* before it is sent to the Control Redistribution (CR) algorithm, based on the known empirical relationship of the estimated versus true parameter. The refined estimate, $\mathbf{a}_{out}$, incorporates knowledge of the true parameter given the estimate of the true parameter, as seen in Figures 24 through 28. Thus, $\mathbf{a}_{out}$ more closely represents $\mathbf{a}_{true}$ than does $\hat{\mathbf{a}}$. Further, logic is incorporated such that CR (assuming only complete failures) or modified CR (assuming partial failures) is applied appropriately, as discussed in Section 3.6.

5.6.2 Results

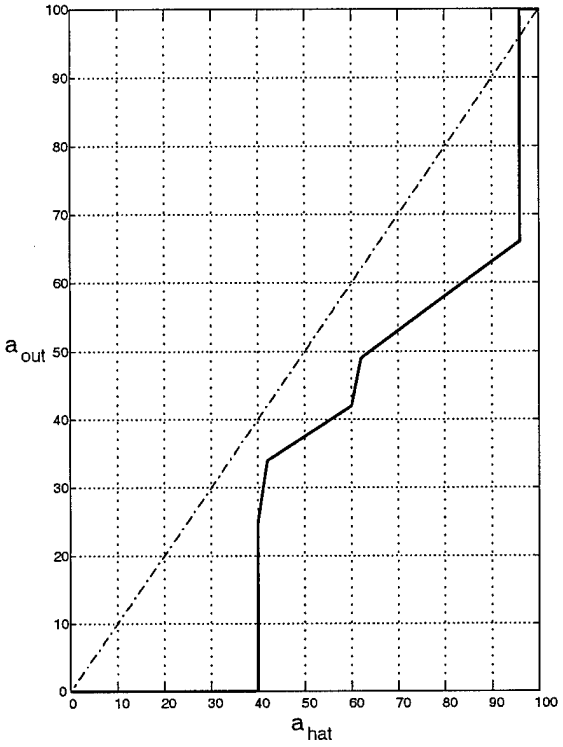
Figures 29 through 33 repeat the $\hat{\mathbf{a}}$ versus $\mathbf{a}_{true}$ plots in subplot (a) and show the $\mathbf{a}_{out}$ versus $\hat{\mathbf{a}}$ plots in subplot (b). The function $\mathbf{a}_{out}(\hat{\mathbf{a}})$ is used in the MMAE with Filter Spawning. When an estimate, $\hat{\mathbf{a}}$, at time t_i is computed by the algorithm, the relationships given in Figures 29 through 33 are used to find the refined estimate, $\mathbf{a}_{out}$, depending on the actuator failure. In words, "Given a parameter estimate, $\hat{\mathbf{a}}$, the refined estimate, $\mathbf{a}_{out}$, is determined and used by CR."

5.6.2.1 Refined Parameter Estimate Versus Parameter Estimate Plots. To see how the mapping given in the "*Refined* Parameter Estimate Versus Parameter Estimate Plots" in the subplot (b) for Figures 29 through 33 is used, consider Figure 29(b):

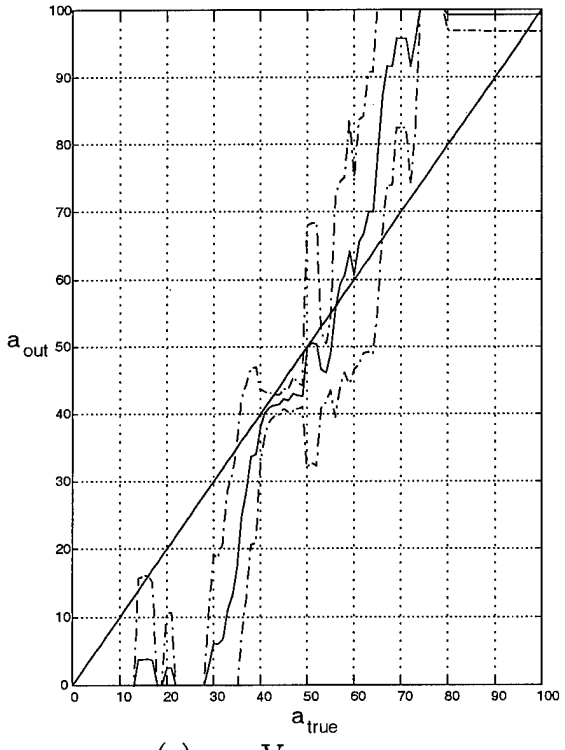
1. If the parameter estimate is less than 40%, then the refined estimate, $\mathbf{a}_{out}$, is considered a complete left stabilator failure estimate. This assures that actuators with small effectiveness values will not receive any control input when applying *unmodified* CR (i.e., for complete failures only).



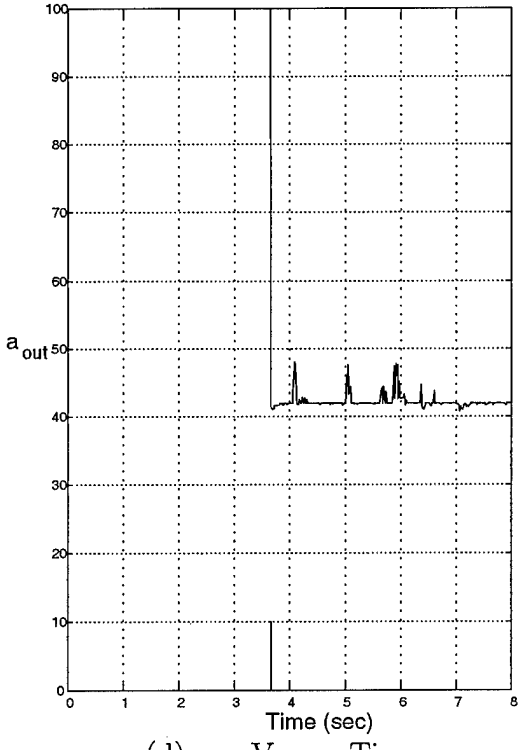
(a) â Versus a_{true}



(b) a_{out} Versus â

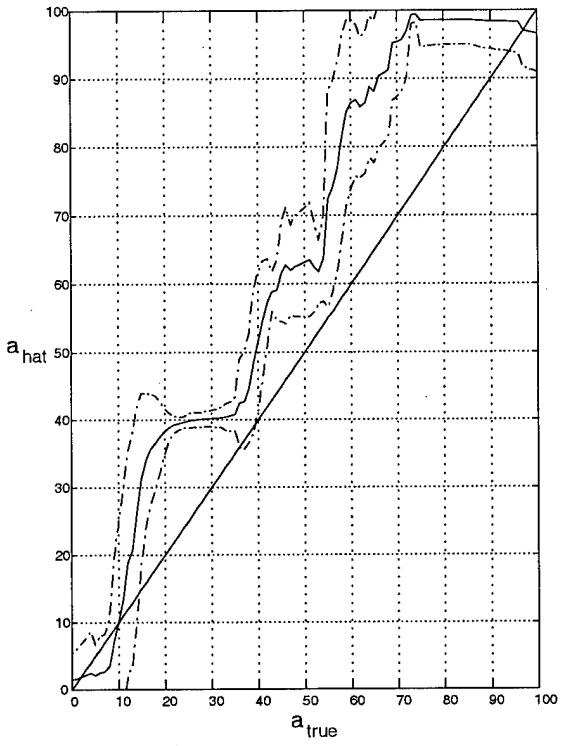


(c) a_{out} Versus a_{true}

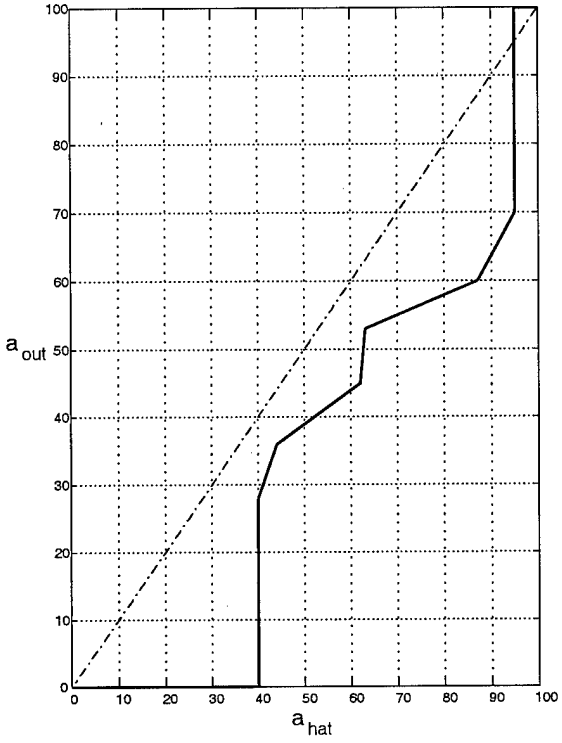


(d) a_{out} Versus Time

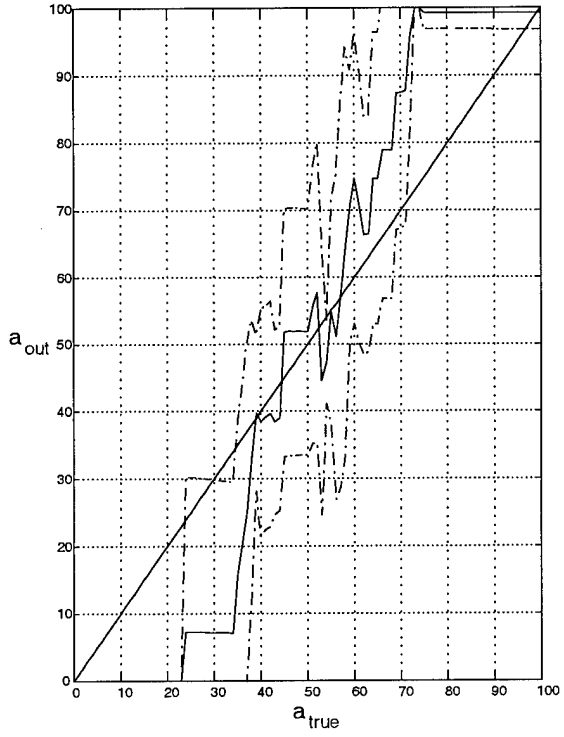
Figure 29. Left Stabilator Failure



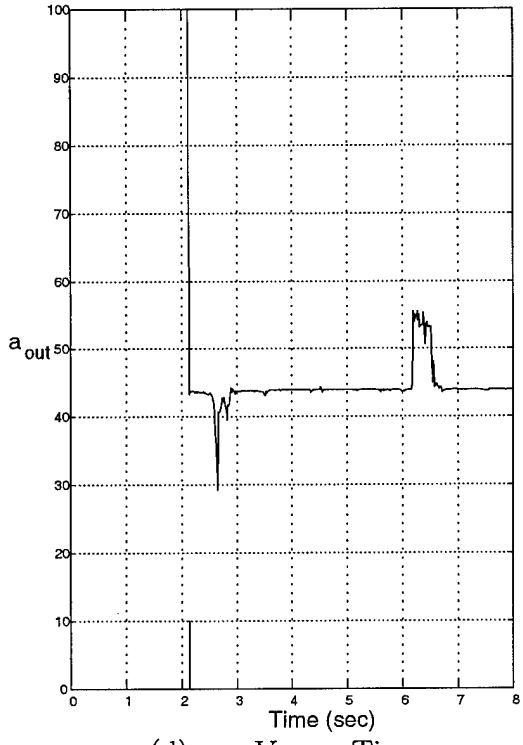
(a) â Versus a_{true}



(b) a_{out} Versus â

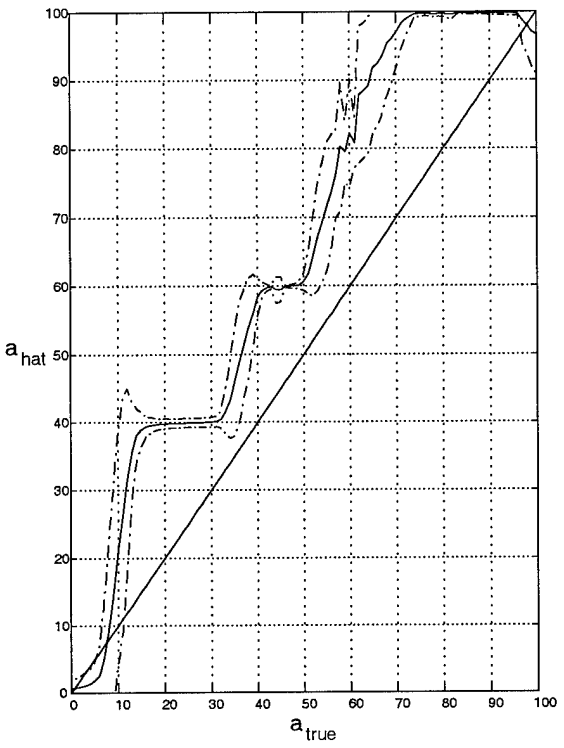


(c) a_{out} Versus a_{true}

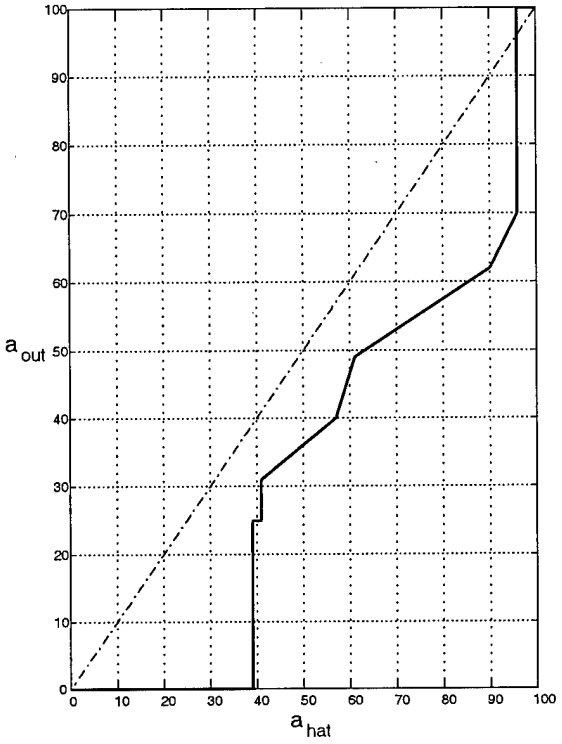


(d) a_{out} Versus Time

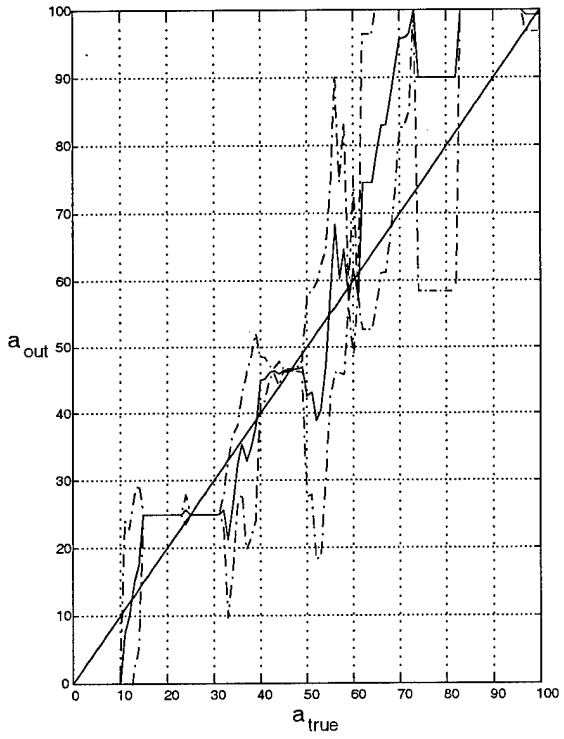
Figure 30. Right Stabilator Failure



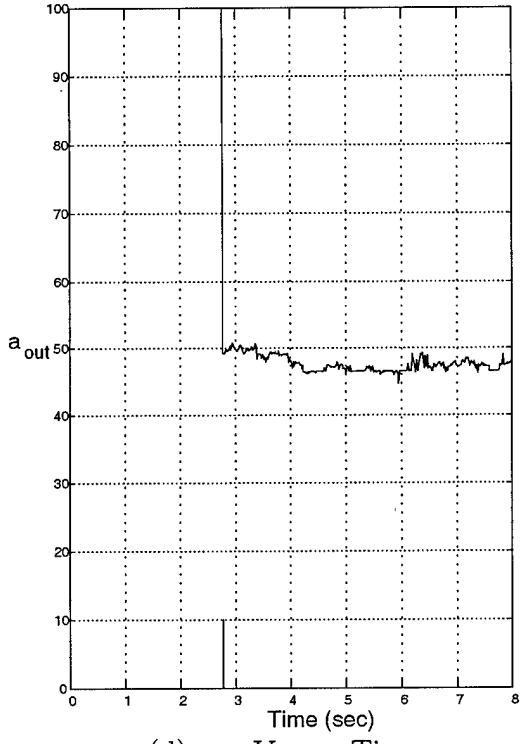
(a) â Versus a_{true}



(b) a_{out} Versus â

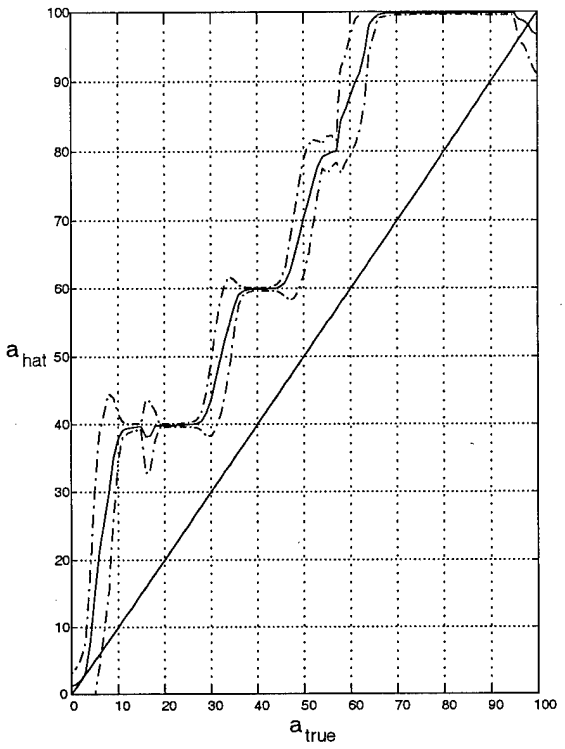


(c) a_{out} Versus a_{true}

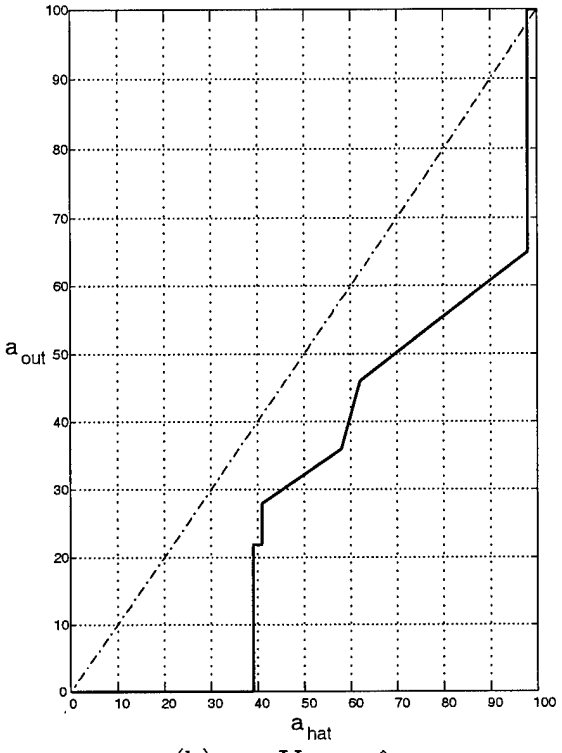


(d) a_{out} Versus Time

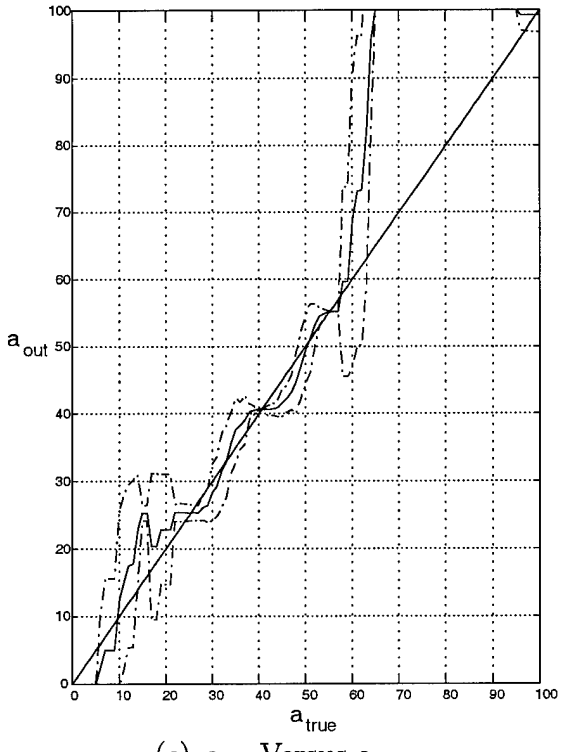
Figure 31. Left Flaperon Failure



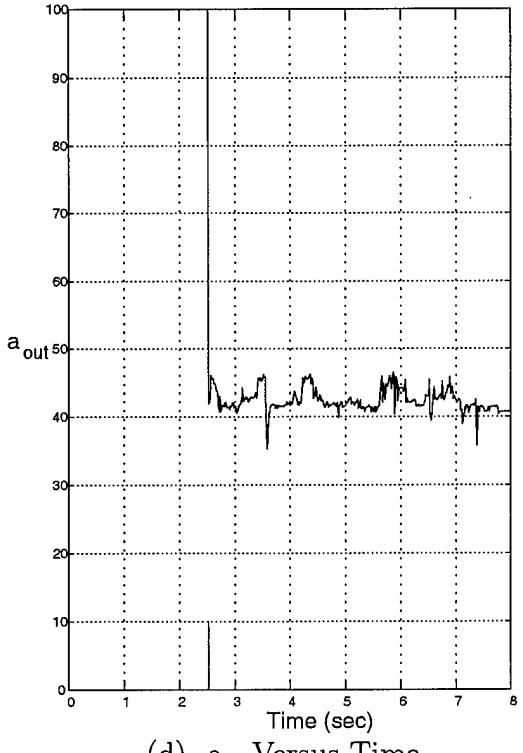
(a) â Versus a_{true}



(b) a_{out} Versus â

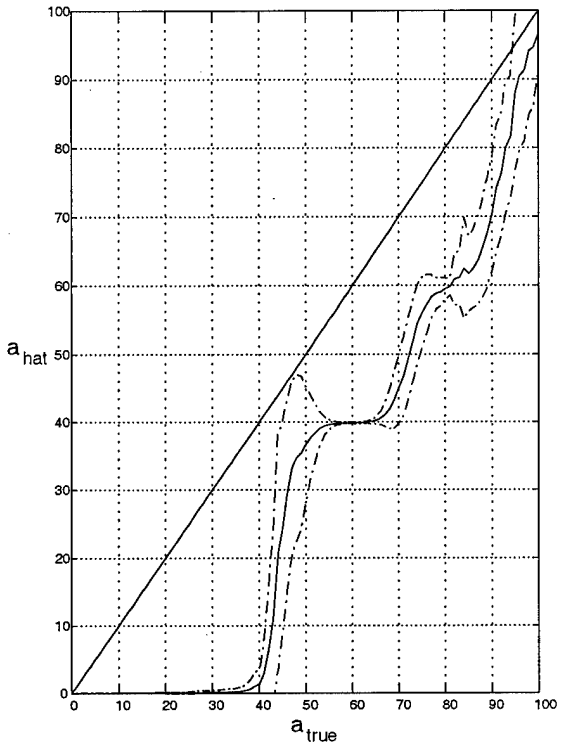


(c) a_{out} Versus a_{true}

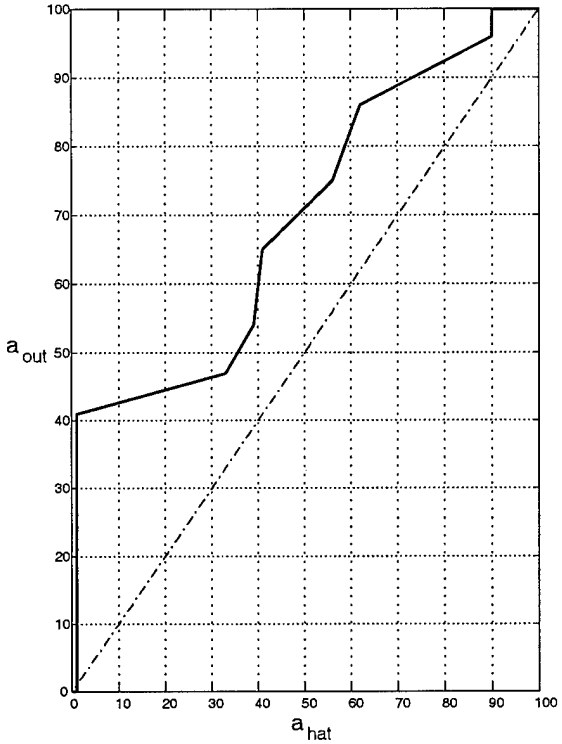


(d) a_{out} Versus Time

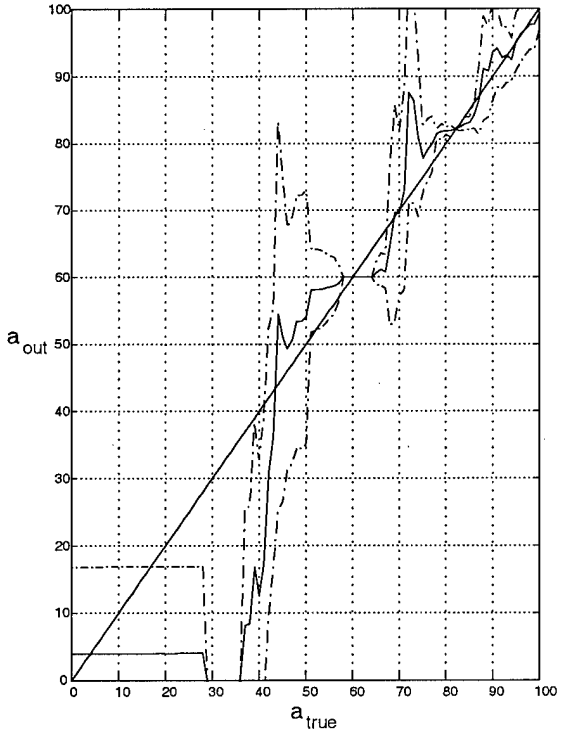
Figure 32. Right Flaperon Failure



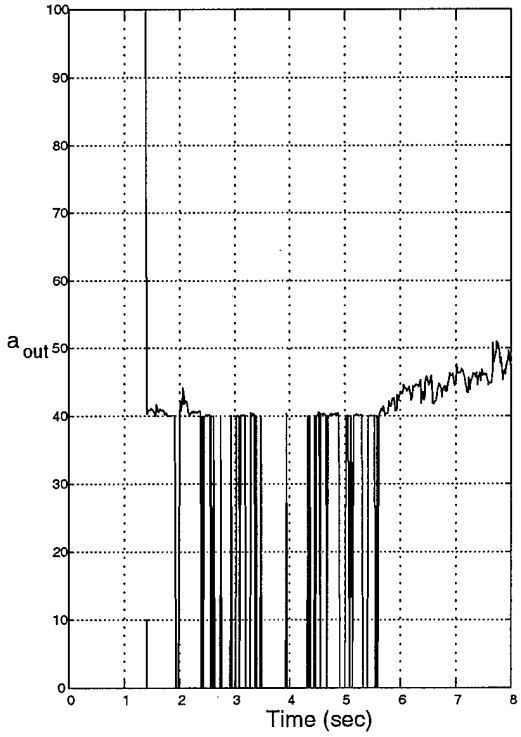
(a) $\hat{a}$ Versus a_{true}



(b) a_{out} Versus $\hat{a}$



(c) a_{out} Versus a_{true}



(d) a_{out} Versus Time

Figure 33. Rudder Failure

2. If the parameter estimate is between 40% and 96%, then the estimate is refined by a piecewise linear relationship to represent the true parameter more appropriately before it is used by *modified* CR. In this region, the result is intended to provide an $\mathbf{a}_{out}$ essentially equal to $\mathbf{a}_{true}$. Notice in Figures 31(b) and 32(b) that $\mathbf{a}_{out}$ levels at 25% and 22%, respectively, over the domain $\mathbf{a}_{true} = [39\%, 41\%]$. This is done to map the $\hat{\mathbf{a}}$ values in the flattened regions in Figures 31(a) and 32(a), respectively, to an appropriate $\mathbf{a}_{out}$ (the "middle" of the flattened region is used).

3. If the parameter estimate is higher than 96%, then the refined estimate is considered a fully functional estimate and control is sent as usual.

Notice that $\mathbf{a}_{true}$ is not a factor for *implementation.* Rather, only the estimate, $\hat{\mathbf{a}}$, is used in the evaluation of $\mathbf{a}_{out}$ and control generation, because $\mathbf{a}_{true}$ is not known. Further, the only deviation between $\mathbf{a}_{out}$ and $\mathbf{a}_{true}$ in the intermediate range denoted as "2." in the list above, is the stochastic nature of the problem *and* the fact that $\mathbf{a}_{out}$ is based on an empirical time-averaged estimate starting two seconds after detection. Thus, when $\hat{\mathbf{a}}(t_i)$ does not match the time-averaged estimate, $\hat{\mathbf{a}}$, some error will exist in the $\mathbf{a}_{out}$ estimate.

5.6.2.2 Refined Parameter Estimate Versus True Parameter and Time Plots. To see how well the refined parameter estimate compares to the true parameter, consider the "*Refined* Parameter Estimate Versus True Parameter Plots" in the subplot (c) for Figures 29 through 33. These plots were formed using the refined parameter estimate given by the MMAE with Filter Spawning algorithm (incorporating the mappings given in the subplot (b) in Figures 29 through 33) at a time *two seconds after the failure was declared.* The plots show the mean as a solid line, and the $\pm 1\sigma$ bounds as "dash-dot" lines symmetrically displaced about the mean, where the mean and standard deviation was computed over 10 Monte Carlo runs for the refined parameter estimate given by the MMAE with Filter Spawning algorithm two seconds after the failure was declared for each run.

A single time point was chosen to compare the refined estimate to the true estimate for a one-to-one correspondence. It is desirable to form a redistributed control vector as soon as possible. As a result, rather than wait to time-average the refined estimate (as was done for the parameter estimate versus time plots), the refined parameter estimate given two seconds after declaration is used to form a redistributed control vector. The two-second point was chosen based on the convergence trends seen in the parameter estimate versus time plots in Figures 19 through 22.

Because choosing a single point upon which to base the plots may bring doubt to the validity of the plots, a representative *single* Monte Carlo run at a single true effectiveness 50% is shown in the subplot (d) in Figures 29 through 33. A true effectiveness of 50% was chosen because it represents one point in the region where an estimate of the effectiveness is sought. (This is in contrast to low true effectiveness values, which the mapping has forced to zero or an indication of complete failure, and high true effectiveness values, which the mapping has forced to one and thus a declaration of a fully functional aircraft.)

The "*Refined* Parameter Estimate Versus Time Plots" shown in the subplots (d) bring out two things not seen in the "*Refined* Parameter Estimate Versus True Parameter Plots" in the subplots (c). First, time of failure declaration is shown with a "tickmark" placed at the bottom of the plot. For instance, in Figure 29(d), the failure is detected at $t_i = 3.6$ sec. Thus, the time to declare a failure for one Monte Carlo run at one true effectiveness, 50%, gives the designer an indication of when a failure is declared. The time at which the refined estimate is used to be compared to the true effectiveness is two seconds after this declaration. Notice that, consistent with previous results, the right channel failures are detected faster than the left channel failures and the flaperons are detected faster than the stabilators for this single Monte Carlo run at a true effectiveness value of 50%. Second, the "*Refined* Parameter Estimate Versus Time Plots" show that the refined parameter estimate remains consistent over time for this single Monte Carlo run at a true effectiveness value of 50%, except for the rudder. The rudder will be addressed in Section 5.6.2.3.

With the explanation of how the "*Refined* Parameter Estimate Versus Time Plots" in the subplots (d) in Figures 29 through 33 are composed, an analysis can be made. First, subplots (c) show that mapping low true effectiveness values to the complete failure estimate is improved over that attainable in subplots (a) with the original estimate. Second, mapping high true effectiveness values to the fully functional estimate is improved. Third, the bias in the middle region for true effectiveness values in subplots (a) is removed in subplots (c). Fourth, the levelling in the mapping in Figure 31(b), as discussed in Section 5.6.2, successfully mapped the true effectiveness values in the range $15\% - 35\%$ to an $\mathbf{a}_{out} = 25\%$. Finally, consistent with previous results, the right stabilator and right flaperon do particularly well.

5.6.2.3 Performance Enhancement.

The refined parameter estimate for the rudder in Figure 33(c) is relatively poor in the region $\mathbf{a}_{true} = 0\% - 35\%$. For instance, a true effectiveness of 0% does not consistently map to a refined parameter estimate of 0%, which is important in deciding to use modified CR (for partial failures) or CR (for complete failures). In addition, the single Monte Carlo run shows the refined estimate "jumping" from 0% to 40%. Because the refined estimate will be used directly by the control scheme, rather than allow "jumping", the refined estimate is mapped to zero. This conservative approach uses a complete failure estimate until the effectiveness estimate can be obtained consistently. (Rather than revising the mapping, additional manipulation could be performed to remove the "jumping" before using the estimate in a control scheme.) This gives insight that the mapping of Figure 33(b) should be reconsidered. The rudder mapping in Figure 33(b) maps:

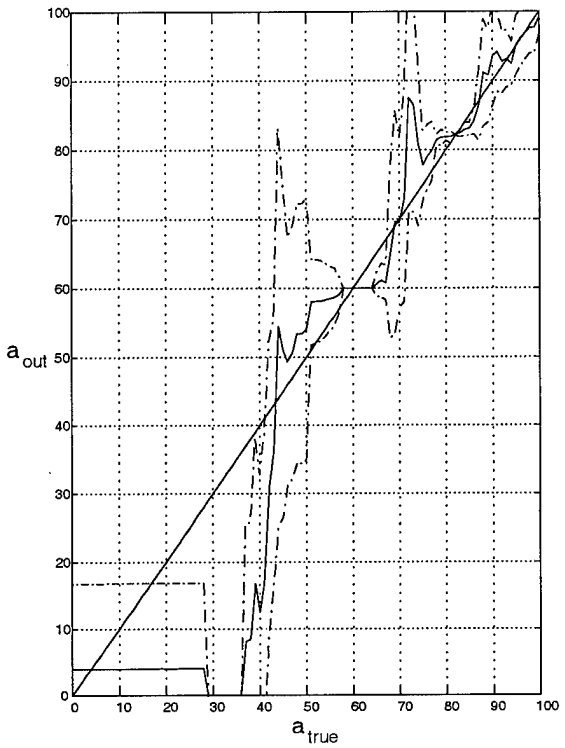
$$\hat{\mathbf{a}} < 1\% \Rightarrow \mathbf{a}_{out} = 0\% \tag{5.5}$$

$$\hat{\mathbf{a}} = 1\% \Rightarrow \mathbf{a}_{out} = 40\% \tag{5.6}$$

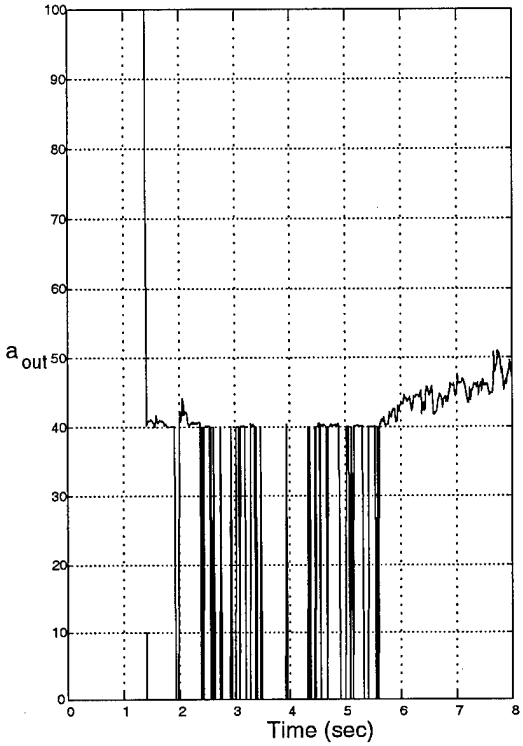
In an attempt to remove the jumping (as seen in the single Monte Carlo run in Figure 33(d) and in the $\pm 1\sigma$ bounds in Figure 33(c) for the region $\mathbf{a}_{true} = 0\% - 30\%$), the mapping is redefined:

$$\hat{\mathbf{a}} < 5\% \Rightarrow \mathbf{a}_{out} = 0\% \tag{5.7}$$

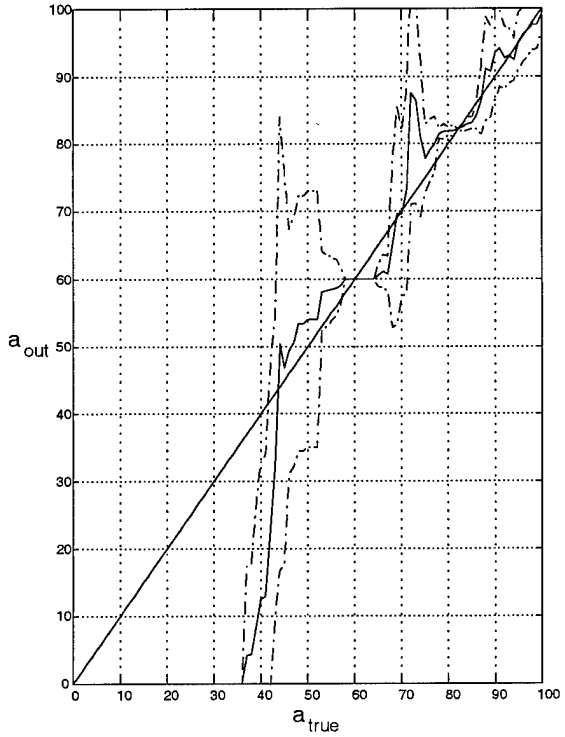
$$\hat{\mathbf{a}} = 5\% \Rightarrow \mathbf{a}_{out} = 40\% \tag{5.8}$$



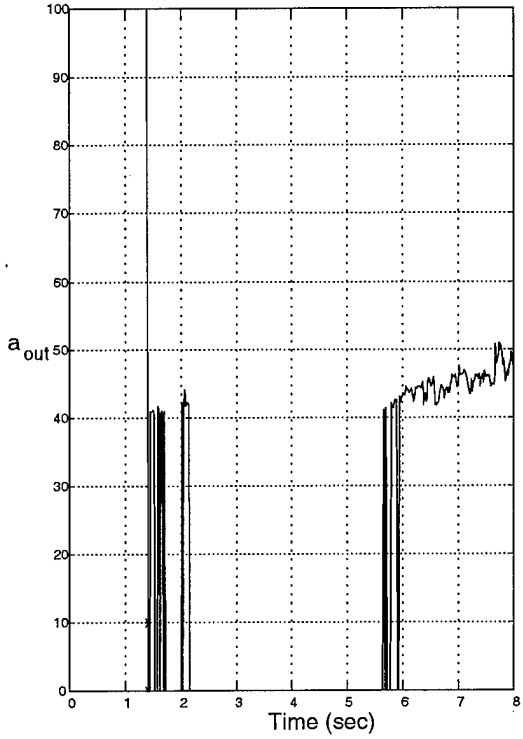
(a) a_{out} Versus a_{true}



(b) a_{out} Versus Time



(c) Alternate Mapping: a_{out} Versus a_{true}



(d) Alternate Mapping: a_{out} Versus Time

Figure 34. Rudder Failure Comparision

In particular, the break-point for the mapping in Figure 33(b) was moved from $\hat{\mathbf{a}}$ =1% to $\hat{\mathbf{a}}$ =5%. The effect is shown in Figure 34, where subplots (a) and (b) are repeated from Figure 33's subplots (a) and (b), respectively, and subplots (c) and (d) are the refined parameter estimate mean$\pm 1\sigma$ versus true parameter plot and the refined parameter estimate versus time plot, respectively. The refined parameter estimate versus time plot in Figure 34(d) is, again, a single Monte Carlo run at a true effectiveness of 50% (the symbol * is placed at the declaration time in subplot (d) for clarity). The jumping is significantly reduced in the refined parameter estimate versus time plot in the range from 2 seconds to 5.5 seconds. After 5.5 seconds, the refined parameter estimate rises near the correct effectiveness value. Thus, the "bouncing" region is forced to zero (and the failure is treated as a complete rudder failure) until the refined estimate attains the correct parameter value. This conservative approach uses a refined parameter estimate of zero when the algorithm produces an indecisive estimate (seen through the bouncing behavior). Thus, the control scheme will treat the failure as a complete failure (and use unmodified CR) until the effectiveness is consistently estimated as a partial failure (and then use modified CR). Further, true effectiveness values in the region 0% to 35% are consistently mapped to a refined estimate 0%, or a complete rudder failure, as desired. (A similar approach could be employed to improve the refined estimate of a left stabilator failure in the region $\mathbf{a}_{true} = 12\% - 20\%$, as shown in Figure 29(c).) In practice, a designer could choose *other* $\mathbf{a}_{out}$ vs. $\mathbf{a}$ curves, but shown here is *one good set* of such curves and an indication of performance sensitivity to alterations in the mapping definition.

5.7 Summary

This section reviews the design and implementation procedures for MMAE with Filter Spawning as well as modification that can be made by the designer.

5.7.1 Design

The following procedure is accomplished *to form* the refined parameter estimate, $\mathbf{a}_{out}$, to be used for Control Redistribution (CR) for flight control in the face of partial or complete actuator

failures and full sensor failures, based on failure detection and estimation based on MMAE with Filter Spawning:

1. Chose a discretization set or sets for the spawned filters. (Section 5.2)

2. Calculate the conditional probabilities for each filter with the MMAE at each time sample while varying the true actuator effectiveness value. (Section 5.3)

3. Using the conditional probabilities and the respective hypothesized parameter values, determine a blended parameter estimate at each time sample. (Section 5.4)

4. Using the time to convergence as seen in the parameter estimate versus time plots, time-average the parameter estimate mean value to obtain one steady state mean estimate for each true parameter value. (Section 5.5)

5. Refine the estimate to incorporate the relationship between the true parameter value and the parameter estimate. Further, incorporate logic into the formation of the refined estimate to apply unmodified CR or modified CR appropriately, depending on the application. (Section 5.6)

5.7.2 Implementation

The following procedure is accomplished *to implement* the refined parameter estimate, $\mathbf{a}_{out}$, in MMAE with Filter Spawning *at each time sample*:

1. Calculate the conditional probabilities for each filter. (Section 5.3)

2. Using the conditional probabilities and the respective hypothesized parameter values, determine a blended parameter estimate. (Section 5.4)

3. Refine the estimate. (Section 5.6)

   A. If detection has occurred less than two seconds prior to the current time, do not "trust" the estimate yet and conservatively apply CR with $\mathbf{a}_{out} = 0\%$.

B. If detection has occurred more than two seconds prior to the current time, "trust" the estimate and apply CR or modified CR to obtain the least degradation in performance as is possible.

5.7.2.1 Design Modification Options. Just as Section 4.4.5 offered design modification options for obtaining an estimate, this section offers such modifications for obtaining a *refined* estimate. (In Section 4.4.5, the modifications affect the computation of the conditional probabilities, failure detection, and failure estimation. In this section, the modifications affect the computation of the refined estimate.) The designer has the following options available for adjustment to enhance the refined estimate:

1. In computing the blended parameter estimate at each time based on the conditional probabilities of each filter and its corresponding effectiveness hypothesis, an additional blending lower bound may be imposed. For this research, the same blending lower bound used for the state estimate is used in forming the parameter estimate, namely, 0.005. (See Section 2.4.1 for the discussion on lower bounds.)

2. Based on the parameter estimate versus time plots, the designer can select the time to start and end the time-average used in the parameter estimate versus true parameter plots, chosen as $t_{start} = 4$ sec. and $t_{end} = 8$ sec. for this research. (Section 5.4 illustrated the effect of changing t_{start}.)

3. Based on the parameter estimate versus true parameter plots, the designer may arrive at mappings used to *refine* the estimate different from those given here. For the rudder, Section 5.6.2.3 illustrated significant differences in the refined output of two *slightly* different mappings. Thus, the most flexibility in MMAE/CR with Filter Spawning is the map chosen to refine the estimate. Quantization could be employed to force the mapping to discrete levels. Section 5.6.2 demonstrated limited quantization for the flaperons in the region $\hat{a}$ =39% – 41%

as shown in Figures 31(b) and 32(b).

4. Based on the parameter estimate versus time plots, the designer can select the time at which the *refined* estimate is given to CR, chosen as two seconds after a failure is declared for this research. More complicated techniques could be employed, such as windowing (averaging the refined estimate over a given number of samples) or a steady state test (waiting until the difference between the present refined estimate and the refined estimate a given number of sample periods earlier, or averaged over a number of earlier sample periods, is within a specified limit). After initially using the refined estimate, the designer must decide whether or not to use the refined estimate as computed at each successive sample period or to use the refined estimate for a given period of time before using a new refined estimate.

The options presented here and in Section 4.4.5 can be explored fully when a control analysis is sought. This research is complete, having *detected* complete and partial actuator failures and obtained an accurate *estimate* of the true effectiveness for partial actuator failures through the use of MMAE with Filter Spawning. While Section 6.3.1 details the issue of applying control for partial actuator impairments, it is mentioned here that, since Stepaniak [41,42] demonstrated CR's ability to restore control in the case of complete failures, modified CR should be able to restore control in the case of partial failures.

5.8 Conclusion

In this chapter, the results of this research have been presented, walking the reader through the process in forming the results. Discretization sets for the spawned filters were chosen in Section 5.2, "probability plots" were described and shown Section 5.3, "parameter estimate versus time plots" were described in Section 5.4, "parameter estimate versus true parameter plots" were described in Section 5.5, "refined parameter estimate plots" were described and shown in Section 5.6, and a summary of the design, implementation, and modification options was given in Section 5.7.

Chapter 6 - Conclusions and Recommendations

6.1 Chapter Overview

This chapter presents a review of this research in Section 6.2. MMAE with Filter Spawning successfully estimated the failure effectiveness in the face of partial actuator failures. Moreover, useful design tools were developed, allowing the designer to notice trends quickly, improve the parameter estimate, and give insight to how the estimate should be used in control redistribution. The recommendations for future research are given in Section 6.3, including the final step of the detection, estimation and control scheme (controlling the aircraft with detected and estimated partial failures); algorithm *enhancement* recommendations (improvements within the current capabilities of the MMAE with Filter Spawning algorithm); and algorithm *extension* recommendations (increasing the capabilities of the MMAE with Filter Spawning algorithm).

6.2 Performance of MMAE with Filter Spawning

6.2.1 Models and Assumptions

For this research, the "real world" is the Variable Stability, In-flight Simulation Test Aircraft (VISTA) F-16. The "truth model" is a full six-degree-of-freedom model including nonlinear equations of motion, fourth order actuator models, the complete Block 40 flight control system, and the aileron-to-rudder interconnect. A reduced order "design model" is chosen as a linear, time-invariant (LTI), 13 state truth model, including first-order actuator models. The "design model" is linearized about the following flight condition: steady level flight at 0.4 Mach at 20,000 feet.

For this research, design models, each based on a different failure hypothesis, are used in a Multiple Model Adaptive Estimation with Filter Spawning algorithm. Failures are introduced into the design model (and truth model) using the *effectiveness factor*,

$$0 \leq \epsilon \leq 1 \tag{6.1}$$

where $\epsilon = 0$ is a "complete failure", $0 < \epsilon < 1$ is a "partial failure", and $\epsilon = 1$ is a "fully functional device". An actuator failure is introduced into the truth model by multiplying the *commanded*

actuator position by ϵ. An actuator failure is introduced into the LTI design model by multiplying the *column* of the input matrix that corresponds to the failed actuator by ϵ. Only complete sensor failures are considered, modeled by the truth model and design model as the loss of the desired noise-free signal portion of the sensor reading while still admitting sensor noise into the measurement. In the design model, it appears as the *row* of the measurement matrix that corresponds to the failed sensor being multiplied by zero.

6.2.2 MMAE with Filter Spawning Algorithm

The MMAE with Filter Spawning algorithm uses 15 filters at any one time. Twelve of the filters' hypotheses are the same throughout the simulation, as given in Table 14. The remaining three filters are "spawned" based on the current actuator with the highest conditional probability. The hypothesis of each spawned filter is a partial failure of that actuator at a specified effectiveness, where each of the three spawned filters uses a different specified effectiveness. Noticing that the fully functional aircraft filter and complete actuator failure filter (for each individual actuator considered) are also associated with the spawned filter hypotheses, the spawned filters resolve the estimate of the failure effectiveness, $\hat{\epsilon}(t_i)$, more precisely by including five discretization points (rather than just two if spawned filters were not included) used in the blended estimate, expressed in terms of effectiveness percent as

$$\epsilon_j = 0\%,\ \epsilon_{j1} = 40\%,\ \epsilon_{j2} = 60\%,\ \epsilon_{j3} = 80\%,\ \epsilon_{ff} = 100\% \tag{6.2}$$

where ϵ_j is the effectiveness hypothesis of the complete actuator failure filter; $\epsilon_{j1}, \epsilon_{j2}$, and ϵ_{j3} are the effectiveness hypotheses of the spawned filters; and ϵ_{ff} is the effectiveness hypothesis of the fully functional aircraft filter index. The effectiveness hypotheses for the spawned filters in Equation (6.2)

Table 14. Twelve Primary Filter in MMAE Bank

Fully Functional Aircraft	Angle of Attack Sensor Failure
Left Stabilator Failure	Pitch Rate Sensor Failure
Right Stabilator Failure	Normal Acceleration Sensor Failure
Left Flaperon Failure	Roll Rate Sensor Failure
Right Flaperon Failure	Yaw Rate Sensor Failure
Rudder	Lateral Acceleration Sensor Failure

are based on the empirical relationship between the parameter estimate and the true estimate, comparing several sets of hypotheses used as the effectiveness hypotheses for the spawned filters in the MMAE bank when forming the parameter estimate.

6.2.3 Methodology Developed

A systematic method has been developed to provide a refined parameter estimate, $\mathbf{a}_{out}$. First, the mean and standard deviation of the parameter estimate at each time, $\hat{\epsilon}(t_i)$, as computed by the MMAE with Filter Spawning over 10 Monte Carlo runs, is plotted versus time for each true effectiveness value from 0% to 100% in 1% increments. Next, the mean and standard deviation of the parameter estimate are time-averaged (with the time-averaging window based on the convergence characteristics seen in the mean and standard deviation of the parameter estimate versus time plots) to obtain one parameter estimate [mean$\pm 1\sigma$] for each true effectiveness value from 0% to 100% in 1% increments. Finally, the relationship found by plotting the parameter estimate versus the true estimate is used to *refine* the estimate, yielding a mapping from $\hat{\mathbf{a}}_{MMAE}$ to $\mathbf{a}_{out}$ for use by Control Redistribution (CR).

6.2.4 Partial Failure Detection and Estimation

The mean$\pm 1\sigma$ values of the refined parameter estimate, $\mathbf{a}_{out}$, versus the true parameter, $\mathbf{a}_{true}$, for each actuator failure is shown in Figures 35 through 39. In order to compare one $\mathbf{a}_{out}(t_i)$ (as it varies over time) for each $\mathbf{a}_{true}$, the plots in Figures 35 through 39 pertain to the single $\mathbf{a}_{out}(t_i)$ given at a time two seconds after the failure was declared. This time selection was chosen based on plots of the parameter estimate versus time. The mean and standard deviation was computed for the $\mathbf{a}_{out}$ given by the algorithm two seconds after the failure declaration for 10 Monte Carlo runs. The mean is shown as a solid line, and $\pm 1\sigma$ bounds are shown as a "dash-dot" lines symmetrically displaced about the mean.

Notice the characteristics of the refined parameter estimate versus the true parameter plots shown in Figures 35 through 39. The refined parameter estimate for low true effectiveness values is a complete actuator failure estimate, so that unmodified CR (for complete failures) is used rather

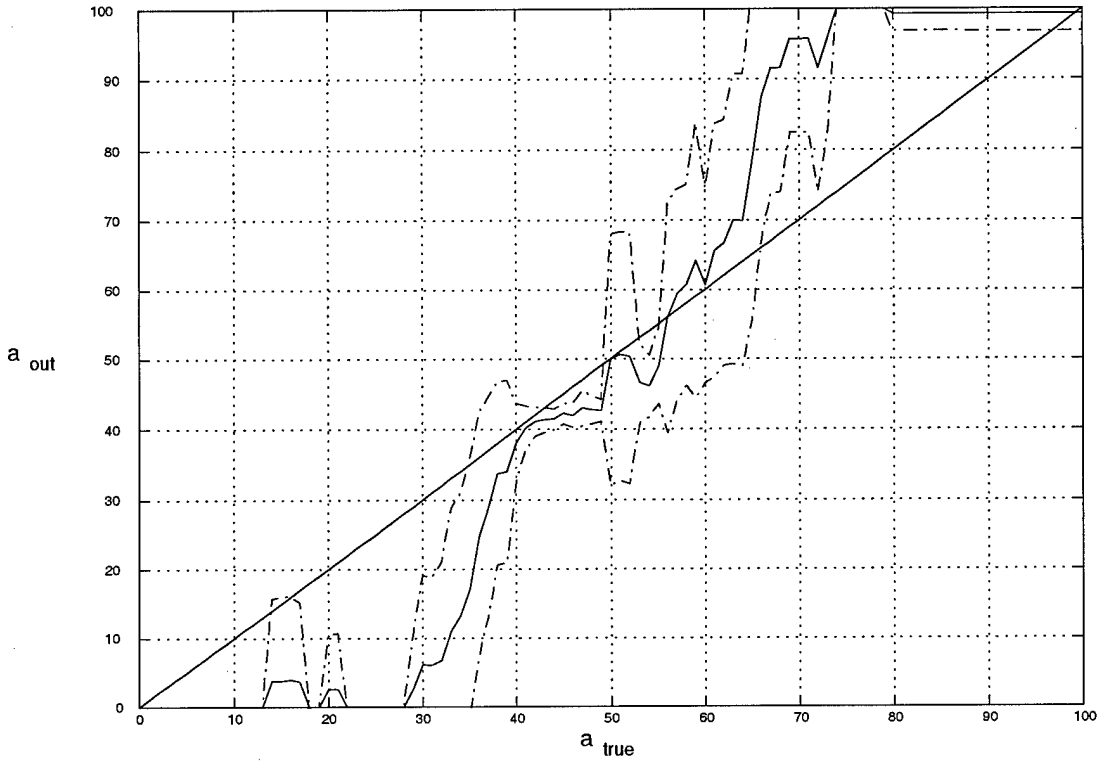


Figure 35. Refined Parameter Estimate Versus True Parameter for a Left Stabilator Failure

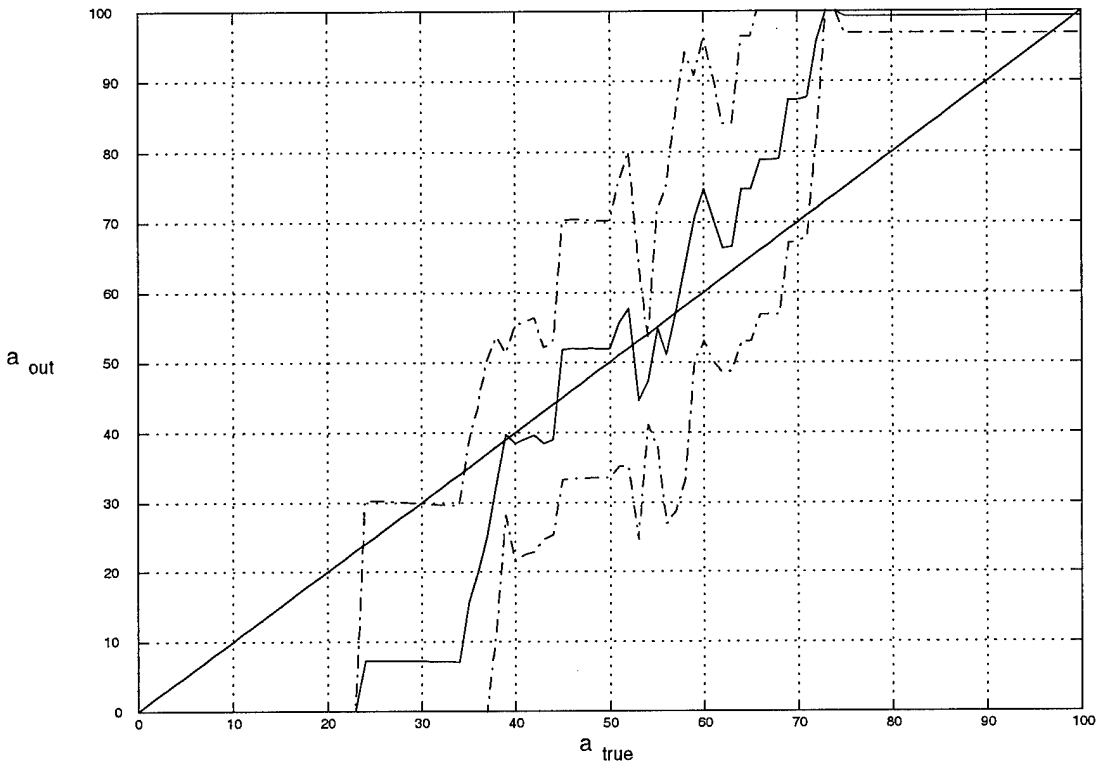


Figure 36. Refined Parameter Estimate Versus True Parameter for a Right Stabilator Failure

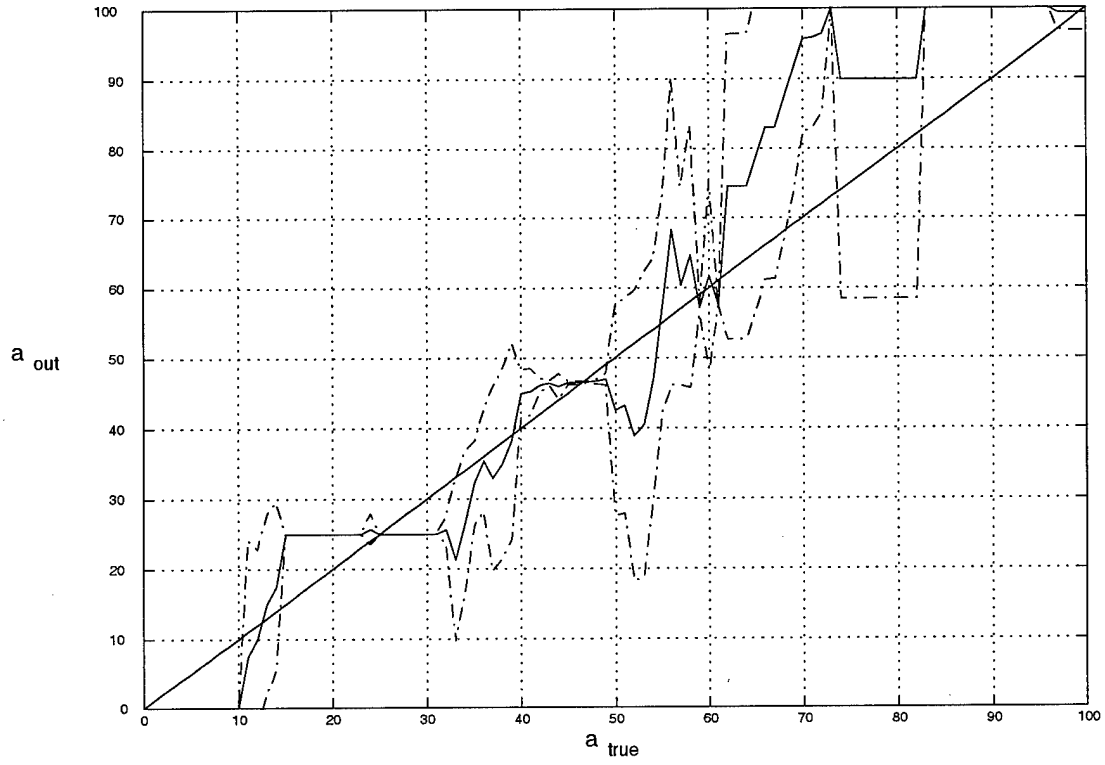


Figure 37. Refined Parameter Estimate Versus True Parameter for a Left Flaperon Failure

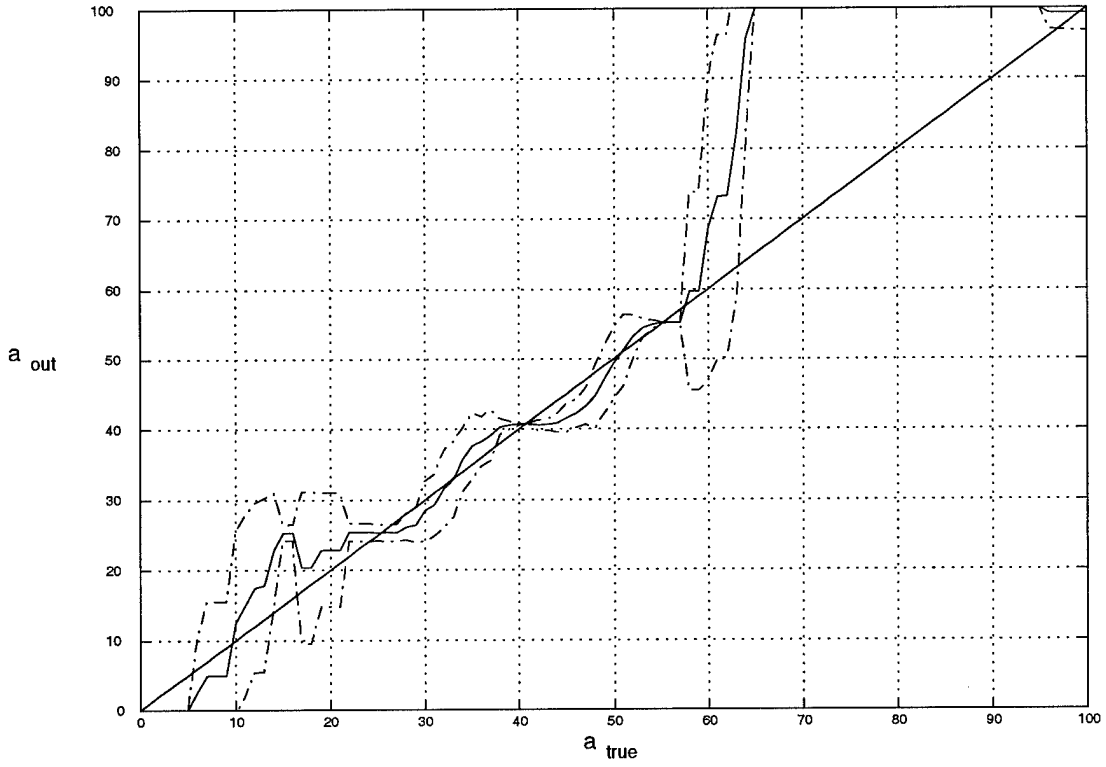


Figure 38. Refined Parameter Estimate Versus True Parameter for a Right Flaperon Failure

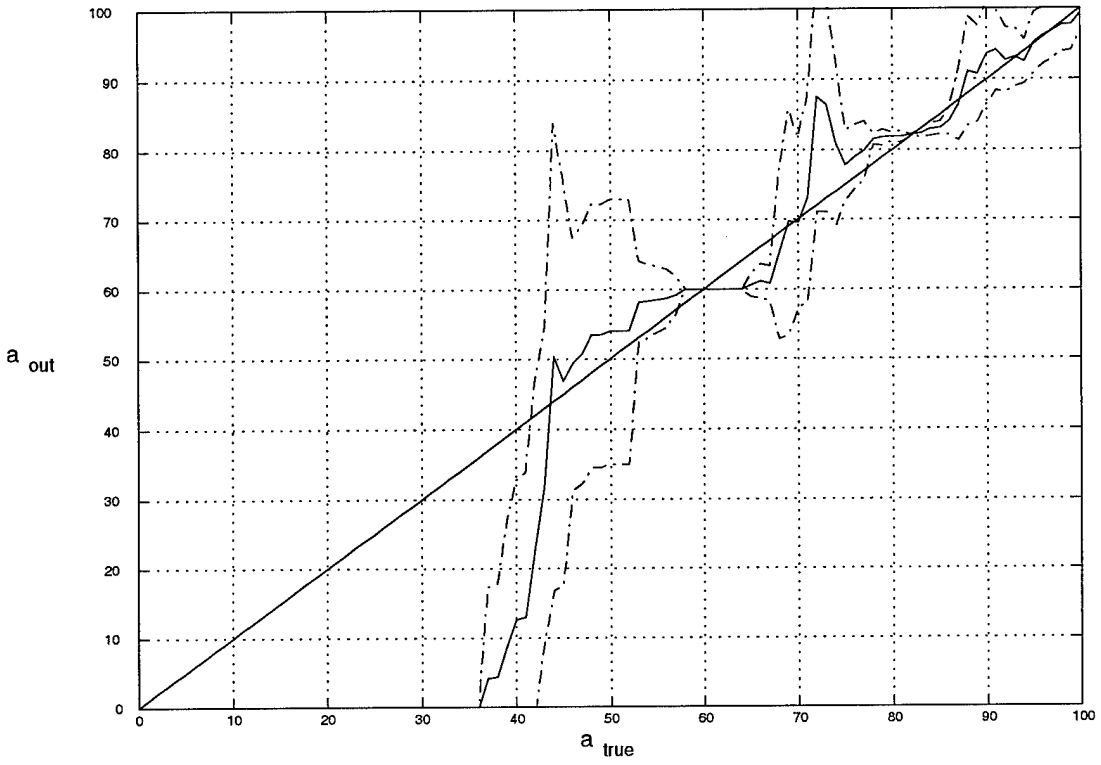


Figure 39. Refined Parameter Estimate Versus True Parameter for a Rudder Failure

than modified CR (for partial failures). Further, the refined parameter estimate for high true effectiveness values is a fully functional aircraft estimate; that is, until the actuator's performance is degraded beyond 30% (or conversely, an effectiveness of less than 70% is attained), the aircraft is declared to be fully functional and the natural robustness of the original Block 40 flight control system is employed to handle such benign partial failures. The exception to this is the refined estimate for a rudder failure, as shown in Figure (39), where a rudder failure is accurately estimated for the slightest degradation (or conversely, for high effectiveness values).

6.3 Recommendations

The performance of the refined estimates for each actuator failure in Figures 35 through 39 demonstrates in a compact manner the ability of MMAE with Filter Spawning to obtain an accurate estimate of complete and partial actuator failures, as applied to the VISTA F-16. This research has concentrated on MMAE with Filter Spawning's ability to *detect* and *estimate* complete and partial actuator failures (as well as complete sensor failures). To complete the application of MMAE/CR with Filter Spawning, *control* should be applied using the refined estimate of the failure's effectiveness. Also, additional recommendations are provided as enhancements to the current algorithm and as expansions beyond detection, estimation, and control as considered in this thesis.

6.3.1 Modified Control Redistribution

Stepaniak [41, 42] showed that, given the correct failure declaration, Control Redistribution (CR) restored performance of an aircraft with failed actuators to nearly that of a fully functional aircraft. While CR has not been demonstrated in the case of partial failures (i.e., *modified* CR), given an accurate estimate of the partial failure's effectiveness, modified CR should restore performance of the aircraft with partially failed actuators to at least that of an aircraft with completely failed actuators and an unmodified CR algorithm. Given an accurate estimate of the partial failure's effectiveness, the partially failed actuator can be used to help control the aircraft; therefore, more control is available in the partial actuator failure case than is available in the complete actuator

failure case. For this reason, given an accurate estimate of the partial failure's effectiveness, *modified* CR should be able to outperform the performance of CR with a complete actuator failure.

The way in which the refined parameter estimate is used by the modified CR algorithm should be explored further. The designer may choose to start using the refined estimate in modified CR after a given number of time samples has passed beyond the declaration of a failure. (This is the suggested method based on this research. It is suggested, based upon the parameter estimate versus time plots, that the refined estimate is used in modified CR two seconds after a failure is declared.) More sophisticated techniques could be employed, such as windowing (averaging the refined estimate over a given number of samples) or a steady state test (waiting until the difference between the present refined estimate and the refined estimate a given number of sample periods earlier, or averaged over a number of prior sample periods, is within a specified limit). Further, the decision must be made whether or not to use the refined estimate as computed at each sample period or to continue to use the refined estimate for a given period of time before using a new refined estimate.

6.3.2 Algorithm Enhancements

The following recommendations are enhancements to the current algorithm. In other words, these enhancements do not expand the capabilities of MMAE with Filter Spawning; rather, they should be used to improve the performance of the current algorithm.

6.3.2.1 Additional Filters. This research used 3 spawned filters. This choice was made recognizing the trade-off between performance and computational loading. More filters should be considered to see if the performance gained merits the additional computational loading. It is important to note that performance is expected to improve with additional spawned filters. An analysis with more filters should be conducted to determine *to what extent* performance is improved. Performance should be sufficiently improved in comparison to the additional computational load required.

6.3.2.2 Multiple Spawned Filters' Hypotheses. This research used a single spawned filter hypothesis set, namely, 40%, 60%, and 80% for all actuators. However, each actuator need not spawn filters with equally discretized effectiveness hypotheses. For example, the partial rudder failure estimation is improved if filters are spawned with hypotheses 25%, 50%, and 75%. For simplicity, all actuators shared the same spawned filter effectiveness hypotheses. Employing different spawned filter effectiveness hypotheses may prove fruitful, and there are no additional memory requirements.

6.3.2.3 Design Modification Options. Throughout this thesis, design modification options have been mentioned. A failure is declared if the conditional probability of an elemental filter exceeds some threshold. For this thesis, the probability threshold for declaring a failure is set to 0.9. Increasing this threshold may result in more undetected failures, while decreasing this threshold may cause more false failure declarations when no failure exists.

When forming a blended parameter estimate using conditional probabilities that have been artificially bounded above zero, a second *blending* lower bound is used to remove hypotheses with conditional probabilities at or near the lower bound. For this research, the blending lower bound used to form the effectiveness estimate is 0.005. This may be adjusted as needed.

For this research, the spawned filters are used in the MMAE bank at all times. The conditional probabilities are recalculated when the spawned filters are not really needed, as in the case of a sensor failure. As an alternative, logic could employ using a bank without spawned filters when the spawned filters are not needed.

For this research, filters are spawned based on the actuator with the highest conditional probability. Deciding when and how to spawn filters could be done differently. For example, the designer may choose to wait until the conditional probability for an actuator failure hypothesis exceeds some limit before spawning filters based upon that actuator.

Based on the parameter estimate versus time plots discussed in Section 6.2.3, the designer can select the time to start and end the time-average used in the parameter estimate versus true param-

eter plots in order to determine the mapping from $\hat{\mathbf{a}}_{MMAE}$ to $\mathbf{a}_{out}$ for use by Control Redistribution (CR). While the start time was chosen as 4 seconds and the end time was chosen as 8 seconds for this research, a designer may wish to explore the results obtained using different time-averaging windows.

Based on the parameter estimate versus true parameter plots discussed in Section 6.2.3, mappings are used to *refine* the estimate for use by Control Redistribution (CR). This is the most powerful design modification option. The results in Figures 35 through 39 are based on one particular mapping to *refine* the estimate for each actuator failure. Another mapping selection may yield vastly different *refined* estimates.

6.3.2.4 Software Implementation. The current MMAE with Filter Spawning algorithm, including the truth model used to validate the design, is implemented using source code written in FORTRAN 77. The compiler uses libraries built into the SunOS 4.1 operating system, and the software cannot link under another operating system. As a result, the current source code for this research must be linked on the SunOS 4.1 operating system only. Further, Sun (Stanford University Networking) no longer supports SunOS 4.1. It is suggested that the dependency of this algorithm's implementation upon the SunOS 4.1 operating system be removed as soon as possible. (For the sake of posterity, notice this is thesis is written 1999, when a fear of the year 2000, or Y2K, exists for operating systems no longer supported by the authors.)

Once moved to a new operating system, the algorithm must be thoroughly verified. That is, the truth model must be validated and all prior research efforts' results must be replicated. For this reason, the transition has been avoided to this date. The following cliché has motivated the delay in this transition: *"If it isn't broke, don't fix it."*

6.3.3 Algorithm Expansions

The following recommendations are *expansions* to the current algorithm. In other words, these enhancements allow MMAE with Filter Spawning to address situations not possible with the MMAE with Filter Spawning algorithm used in this research. These expansions will complete the

capabilities needed for a flight-worthy algorithm. With the results demonstrated in this thesis, the pursuit of a flight-worthy algorithm is, in fact, justified.

6.3.3.1 Flight Envelope Expansion. The current algorithm uses elemental filter design models linearized about the flight condition of 0.4 Mach at 20,000 feet. Thus, as the aircraft deviates from this flight condition, the filters' design models do not match the truth model (or the "real world"). The simulation considered in this thesis has been steady level flight for a duration of only eight seconds, so that the aircraft would not change flight conditions considerably. Strong maneuvers (through pilot commands) and long time durations could not be considered.

Gain scheduling could be used in the design models to adapt to the changing flight condition. The VISTA F-16 Block 40 flight control system uses gain scheduling based on flight condition, so the necessary information is readily available. Further, the redistribution matrix is also a function of the flight condition; thus, the redistribution matrix must be either computed on-line or computed a-priori and stored in memory for regions of the flight envelope.

6.3.3.2 Dual Failures. The possibility exists for more than one failure to occur during flight. When the flight envelope is expanded and the simulation can run for longer periods of time, the possibility for dual failures should be considered. Lewis [18] has demonstrated the ability of MMAE to detect dual, *complete* failures. Dual failures should also be considered in which either failure may be a *partially* failed actuator. With an accurate refined parameter estimate for a partial actuator failure, a second failure declaration can be sought by employing the hierarchical structure developed by Stevens [28, 43] and implemented by Lewis for complete, dual failures. Clark [8] demonstrated that an accurate effectiveness estimate of the first partial actuator failure is essential before a second failure can be sought using the hierarchical structure.

Lewis [18] did find that a second failure was more difficult to detect when the first failure was an actuator, because redistributing dither through CR caused the magnitude of the dither to be reduced in some channels (due to destructive interference). In the case of a first partial actuator,

however, redistributing dither through *modified* CR should provide better excitation than that of unmodified CR because some control authority is maintained by the partially failed actuator.

6.3.3.3 Other Failure Types. For this research, actuator failures other than "failure to free stream" should be considered. The actuator failure models (on which the elemental filters in the MMAE are based) hypothesize actuator failures as a "failure to free stream". The failure is introduced into the truth model in the same fashion. The performance of the MMAE with Filter Spawning should be explored when the failure is introduced into the truth model in a fashion *different* from that hypothesized by the failure models used in forming the elemental filters. In particular, stuck actuators or a deterioration in the actuator's time response should be introduced into the truth model. A complete failure mode analysis would indicate the most crucial forms of failure to consider. While data indicating which type of failure (i.e., failure to free stream, a stuck actuator, or a deterioration in the actuator's time response) is most likely to occur in the real world may avoid the necessity of such an analysis, it serves as a measure of the algorithm's "robustness" to the failure type. This effort might also result in additional elemental filters within the MMAE structure, based upon new models of important failure types that warrant inclusion in order to provide good detection and estimation.

6.4 Chapter Summary

This chapter has reviewed the results of using MMAE with Filter Spawning to obtain an estimate of the effectiveness in the face of partial actuator failures. The review has included the models used to implement MMAE with Filter Spawning, the method developed to refine the estimate, and the refined parameter estimate [mean$\pm 1\sigma$] performance versus the true parameter plots for each actuator failure. Further, recommendations have been given to *enhance* the capabilities of the current algorithm as well as *extend* the capabilities of the algorithm.

APPENDIX A - VISTA F-16 Characteristics

This appendix details the plant characteristics of the VISTA F-16. The model is detailed in Section 3.4.1. The stability derivatives for the plant matrix, **A**, the input matrix, **B**, and the noise injection matrix, **G**, are found by linearizing about steady, level flight at 0.4 Mach at 20,000 feet. The matrices are repeated here. The values of the longitudinal and lateral stability derivatives for this design condition are given in Tables 15 and 16, respectively.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ X'_\theta & X'_u & X'_\alpha & X'_q & 0 & 0 & 0 & 0 \\ Z'_\theta & Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 & 0 \\ M'_\theta & M'_u & M'_\alpha & M'_q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \phi'_r \\ 0 & 0 & 0 & 0 & Y'_\phi & Y'_\beta & Y'_p & Y'_r \\ 0 & 0 & 0 & 0 & 0 & L'_\beta & L'_p & L'_r \\ 0 & 0 & 0 & 0 & 0 & N'_\beta & N'_p & N'_r \end{bmatrix} \tag{6.3}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ X'_{\delta_e} & 0 & X'_{\delta_f} & 0 & 0 \\ Z'_{\delta_e} & 0 & Z'_{\delta_f} & 0 & 0 \\ M'_{\delta_e} & 0 & M'_{\delta_f} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & Y'_{\delta_{dt}} & 0 & Y'_{\delta_a} & Y'_{\delta_r} \\ 0 & L'_{\delta_{dt}} & 0 & L'_{\delta_a} & L'_{\delta_r} \\ 0 & N'_{\delta_{dt}} & 0 & N'_{\delta_a} & N'_{\delta_r} \end{bmatrix} \tag{6.4}$$

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ X'_u & X'_\alpha & X'_q & 0 & 0 & 0 \\ Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 \\ M'_u & M'_\alpha & M'_q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y'_p & Y'_\beta & Y'_r \\ 0 & 0 & 0 & L'_p & L'_\beta & L'_r \\ 0 & 0 & 0 & N'_p & N'_\beta & N'_r \end{bmatrix} \tag{6.5}$$

With the plant matrix known, the open-loop poles can be found as the roots that satisfy the characteristic equation,

$$det\,(s\mathbf{I} - \mathbf{A}) = 0 \tag{6.6}$$

The poles for the plant at this design condition is given in Table 17. Notice that the longitudinal pole, p_3, is *unstable*. This makes the control of this aircraft particularly interesting; without a controller in the loop, the system will diverge.

Table 15. Longitudinal Stability Derivatives

$X'_\theta = -3.1677 \cdot 10^{1}$	ft/sec^2	$Z'_\theta = -1.4158 \cdot 10^{-2}$	1/sec	$M'_\theta = 6.4212 \cdot 10^{-4}$	1/sec
$X'_u = 2.4312 \cdot 10^{-3}$	1/sec	$Z'_u = -1.9598 \cdot 10^{-4}$	1/ft	$M'_u = -1.3465 \cdot 10^{-3}$	1/ft·sec
$X'_\alpha = 1.6353 \cdot 10^{1}$	ft/sec^2	$Z'_\alpha = -4.4041 \cdot 10^{-1}$	1/sec	$M'_\alpha = 1.5251 \cdot 10^{0}$	1/sec^2
$X'_q = -7.3459 \cdot 10^{1}$	ft/sec	$Z'_q = -9.9720 \cdot 10^{-1}$		$M'_q = -5.2692 \cdot 10^{-1}$	1/sec
$X'_{\delta_e} = 2.0878 \cdot 10^{0}$	ft/sec^2	$Z'_{\delta_e} = -6.8439 \cdot 10^{-2}$	1/sec	$M'_{\delta_e} = -3.6448 \cdot 10^{0}$	1/sec^2
$X'_{\delta_f} = -5.4258 \cdot 10^{-1}$	ft/sec^2	$Z'_{\delta_f} = -3.3787 \cdot 10^{-2}$	1/sec	$M'_{\delta_f} = 2.8568 \cdot 10^{-1}$	1/sec^2

Table 16. Lateral Stability Derivatives

$Y'_\phi = 7.7599 \cdot 10^{-1}$	1/sec			$\phi'_r = 1.8245 \cdot 10^{-1}$	
$Y'_\beta = -1.0988 \cdot 10^{-1}$	1/sec	$L'_\beta = -1.8528 \cdot 10^{1}$	1/sec^2	$N'_\beta = 2.8330 \cdot 10^{0}$	1/sec^2
$Y'_p = 1.8084 \cdot 10^{-1}$		$L'_p = -1.5559 \cdot 10^{0}$	1/sec	$N'_p = -4.3291 \cdot 10^{-2}$	1/sec
$Y'_r = -9.9763 \cdot 10^{-1}$		$L'_r = 4.1348 \cdot 10^{-1}$	1/sec	$N'_r = -2.8217 \cdot 10^{-1}$	1/sec
$Y'_{\delta_{dt}} = 1.3771 \cdot 10^{-2}$	1/sec	$L'_{\delta_{dt}} = -8.9904 \cdot 10^{0}$	1/sec^2	$N'_{\delta_{dt}} = -1.0345 \cdot 10^{0}$	1/sec^2
$Y'_{\delta_a} = 5.6951 \cdot 10^{-4}$	1/sec	$L'_{\delta_a} = -1.2407 \cdot 10^{1}$	1/sec^2	$N'_{\delta_a} = -1.3071 \cdot 10^{-1}$	1/sec^2
$Y'_{\delta_r} = 1.6959 \cdot 10^{-2}$	1/sec	$L'_{\delta_r} = 2.8629 \cdot 10^{0}$	1/sec^2	$N'_{\delta_r} = -1.1652 \cdot 10^{0}$	1/sec^2

Table 17. Open-Loop Eigenvalues

Longitudinal	Lateral
$p_{1,2} = -0.0268 \pm j0.1376$	$p_{1,2} = -0.3365 \pm j2.4071$
$p_3 = +0.8341$	$p_3 = -0.0444$
$p_4 = -1.7455$	$p_4 = -1.2305$

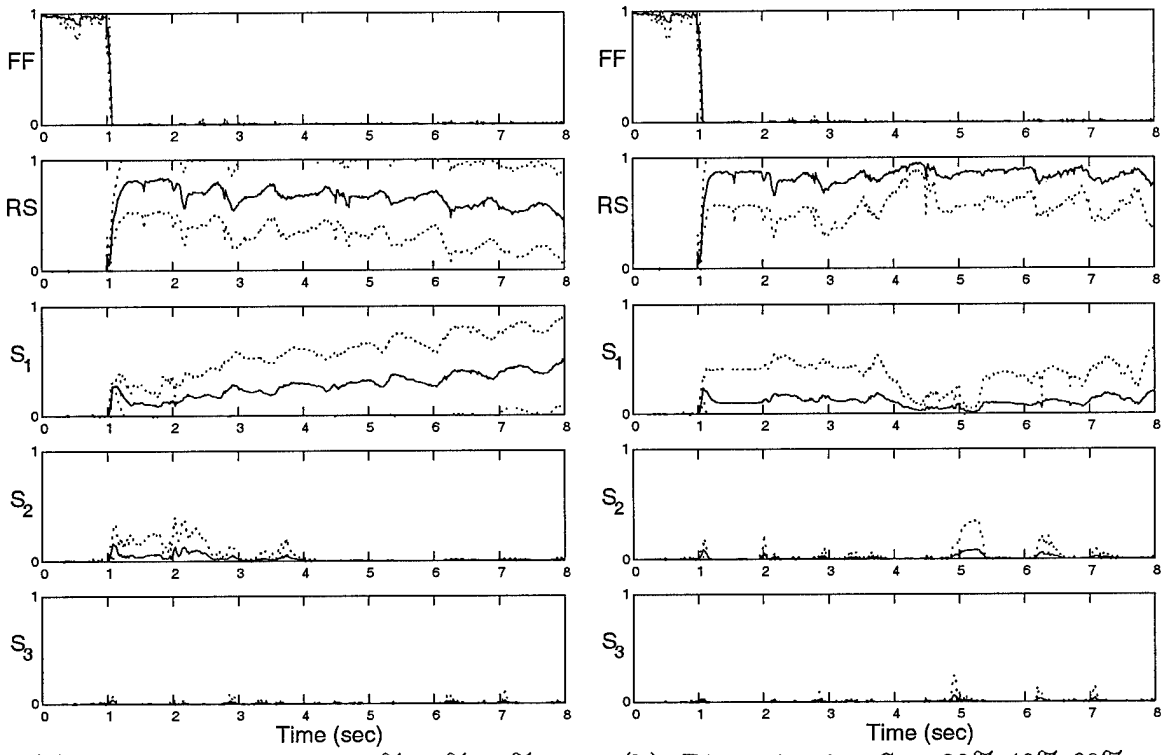
APPENDIX B - Probability Plots

This appendix contains the probability plots for the right stabilator, left flaperon, right flaperon, and the rudder. Recall that the left stabilator probability plots were provided in Section 5.3. The effectiveness is varied according to Table 11, repeated here as

(a)	Discretization Set: 10%, 25%, 50%	Set #1
(b)	Discretization Set: 20%, 40%, 60%	Set #2
(c)	Discretization Set: 25%, 50%, 75%	Set #3
(d)	Discretization Set: 40%, 60%, 80%	Set #4

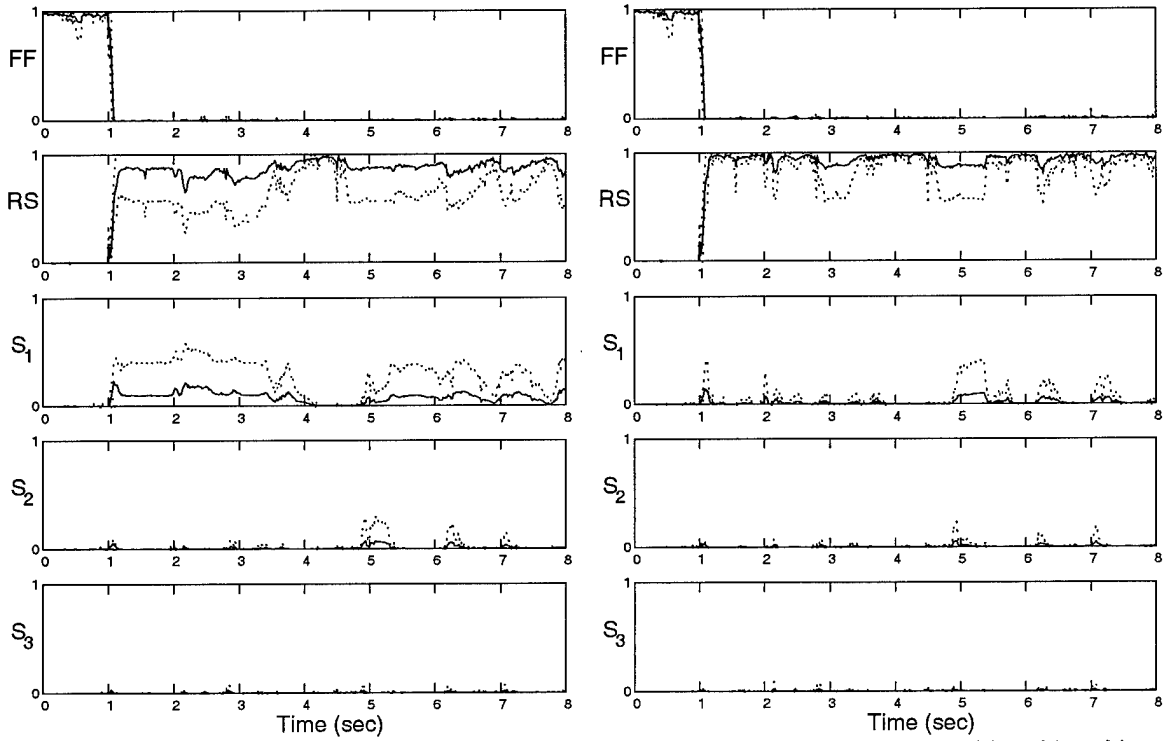
B.1 Right Stabilator

The corresponding figures for each effectiveness value of the right stabilator failure are given as



(a) Discretization Set: 10%,25%,50%

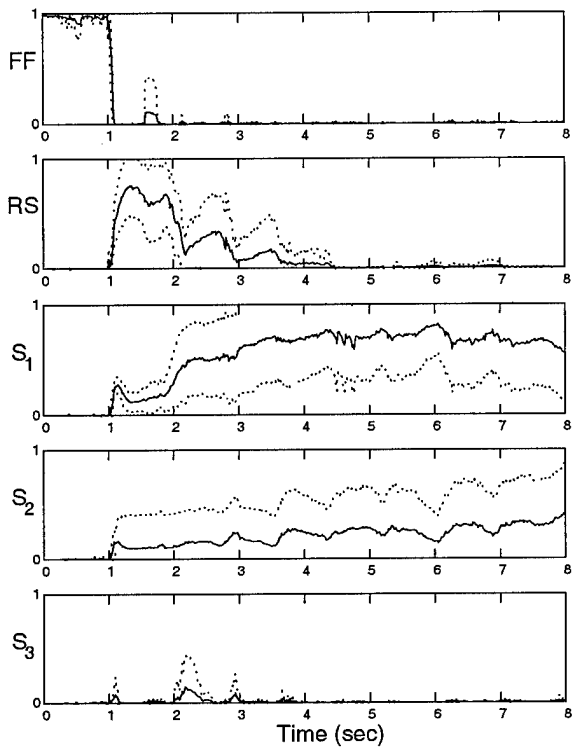
(b) Discretization Set: 20%,40%,60%



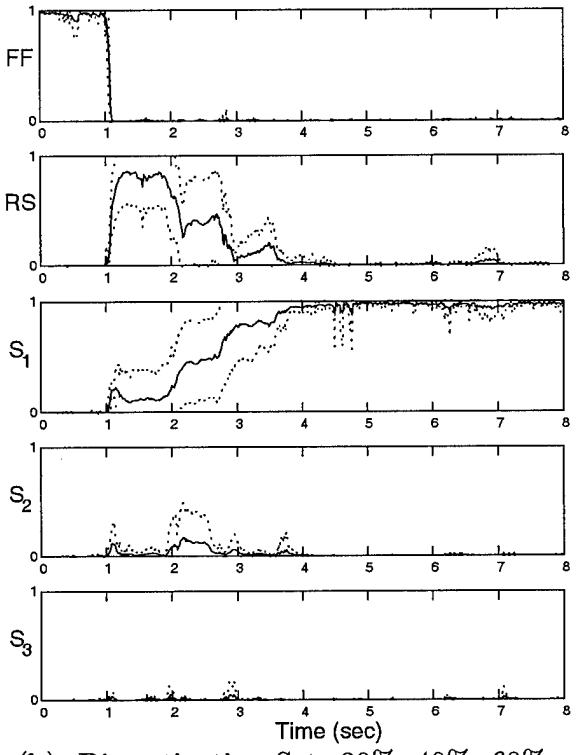
(c) Discretization Set: 25%,50%,75%

(d) Discretization Set: 40%,60%,80%

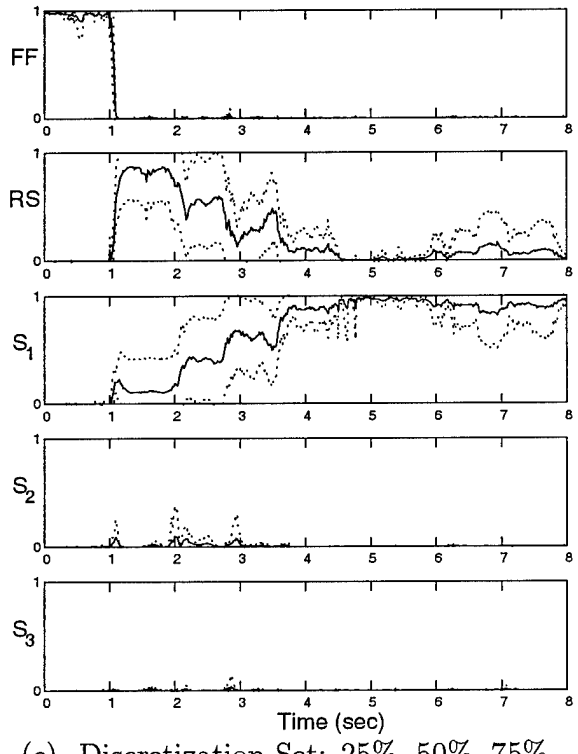
Figure 40. Probability Plot: Right Stabilator Failure, $\epsilon = 0\%$



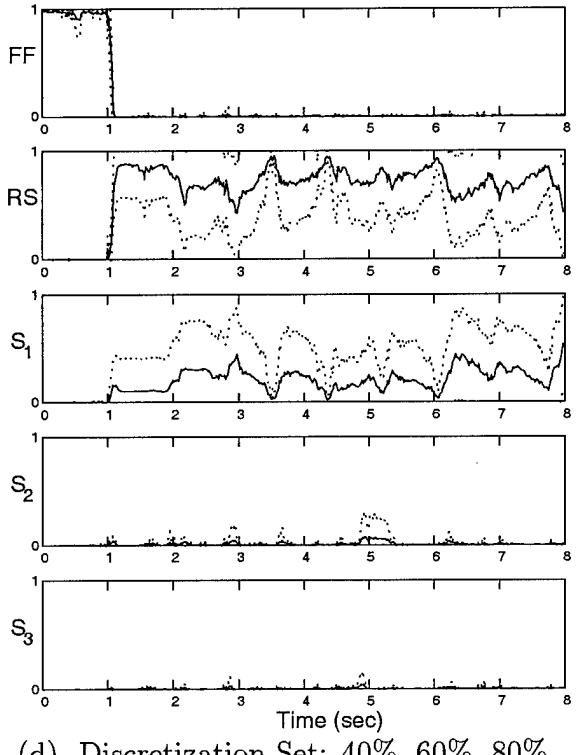
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

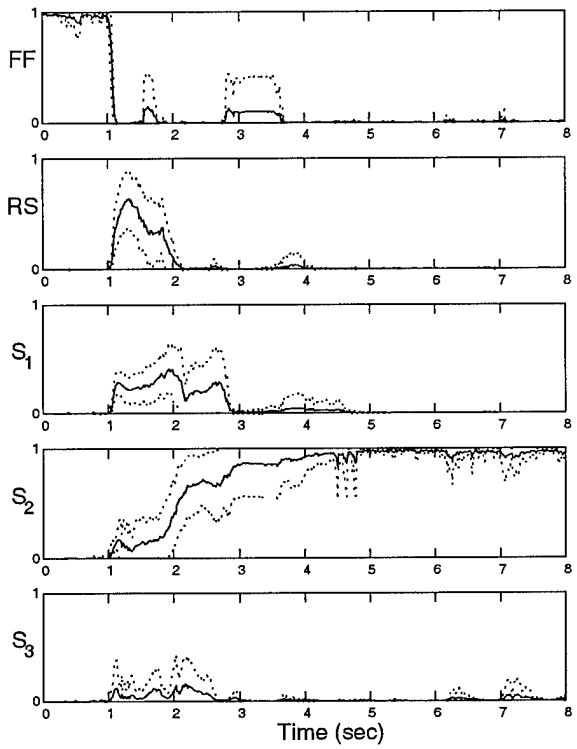


(c) Discretization Set: 25%, 50%, 75%

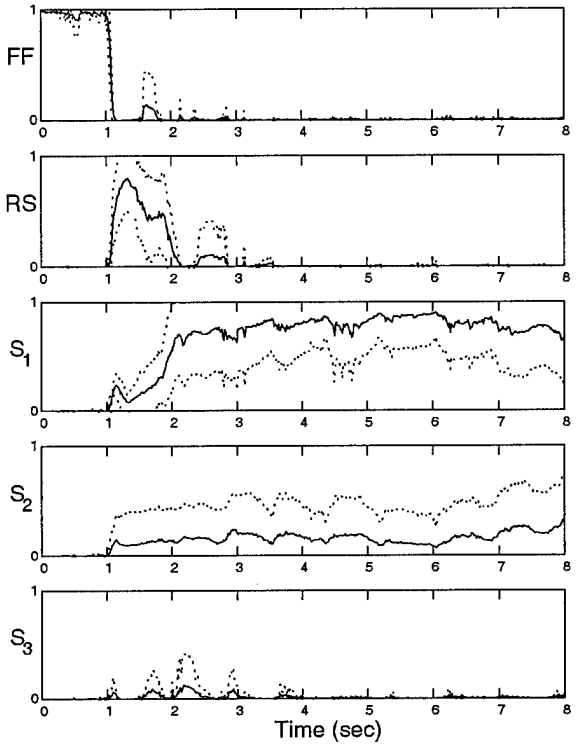


(d) Discretization Set: 40%, 60%, 80%

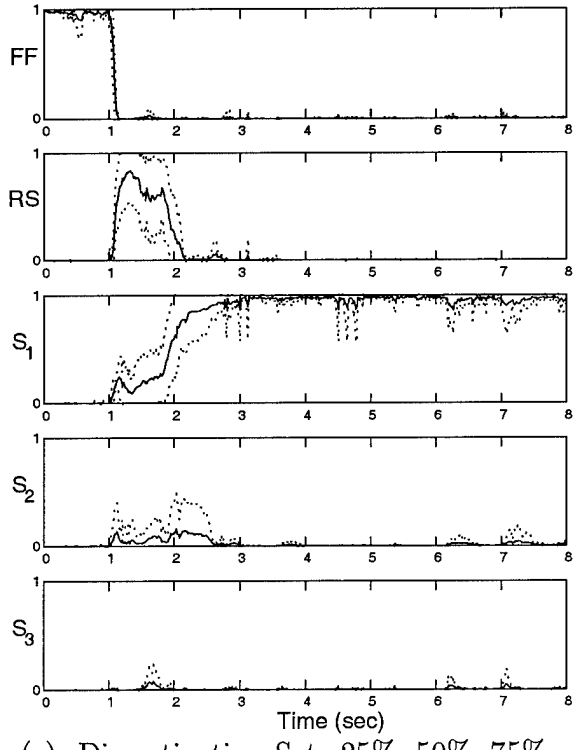
Figure 41. Probability Plot: Right Stabilator Failure, $\epsilon = 10\%$



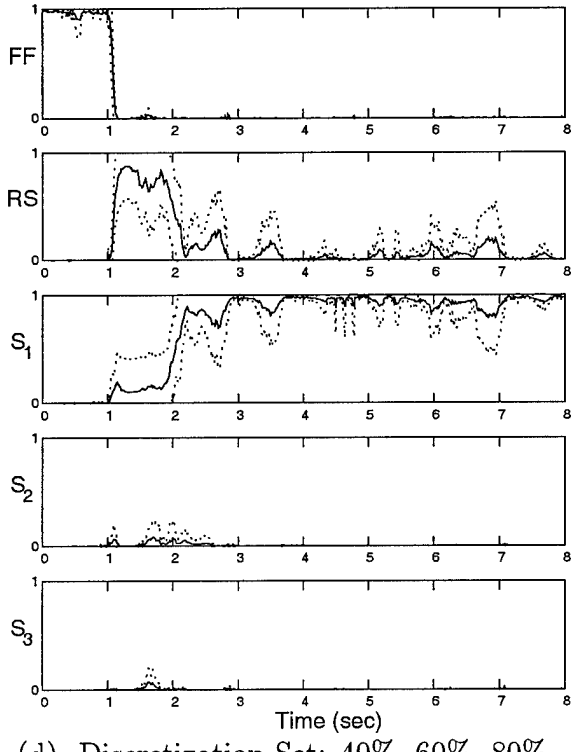
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

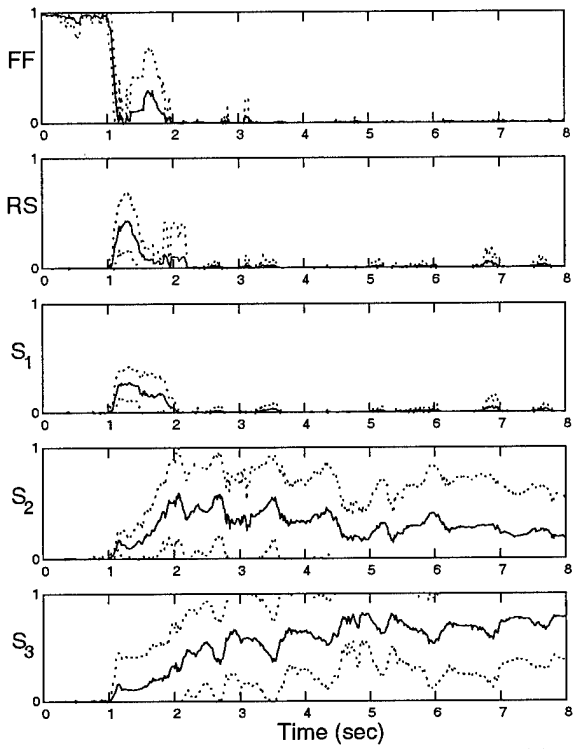


(c) Discretization Set: 25%, 50%, 75%

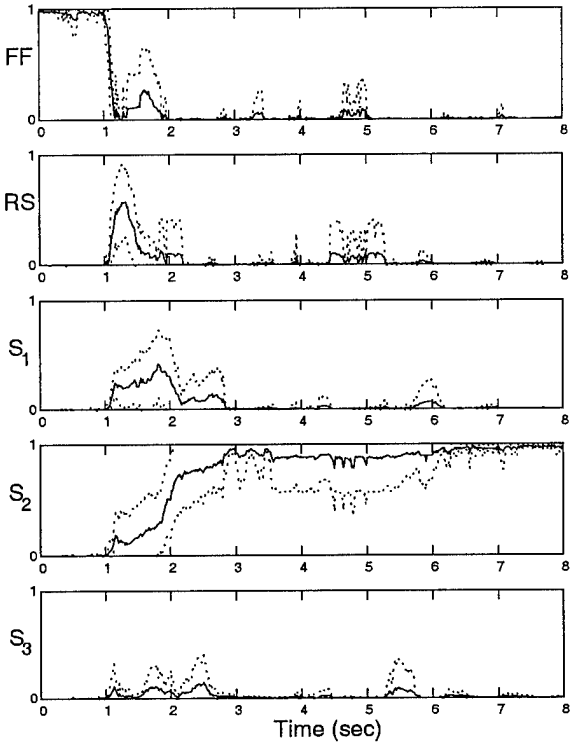


(d) Discretization Set: 40%, 60%, 80%

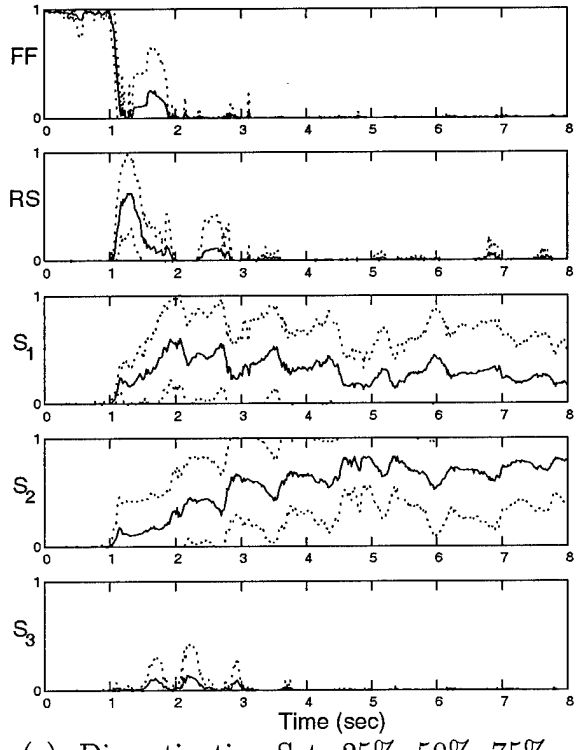
Figure 42. Probability Plot: Right Stabilator Failure, $\epsilon = 20\%$



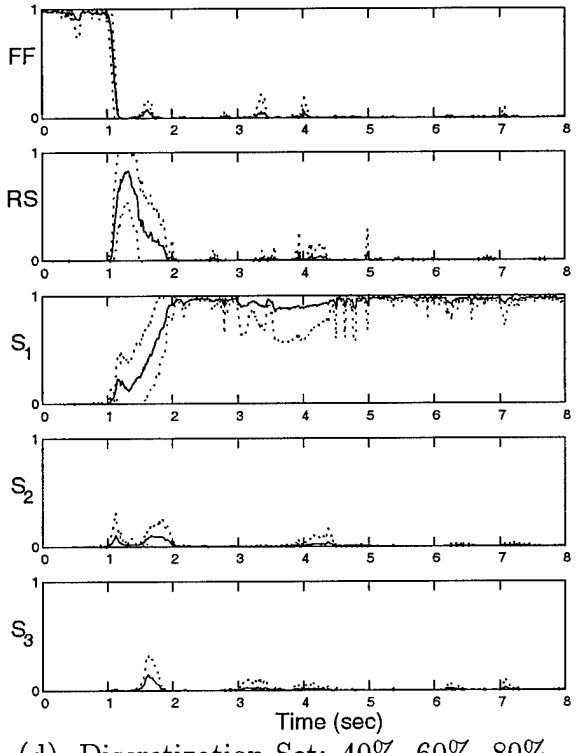
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

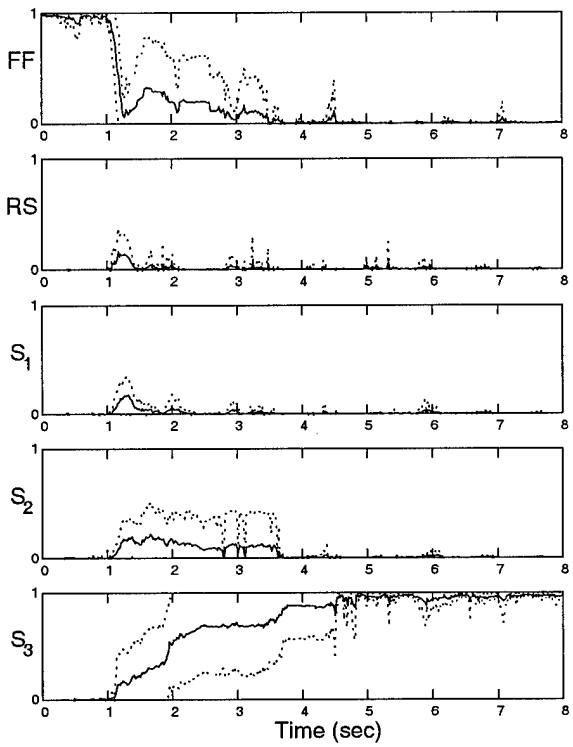


(c) Discretization Set: 25%, 50%, 75%

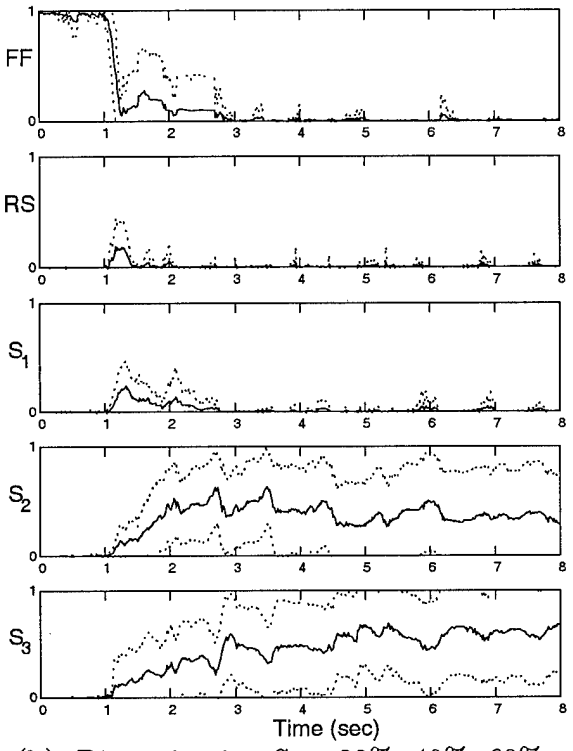


(d) Discretization Set: 40%, 60%, 80%

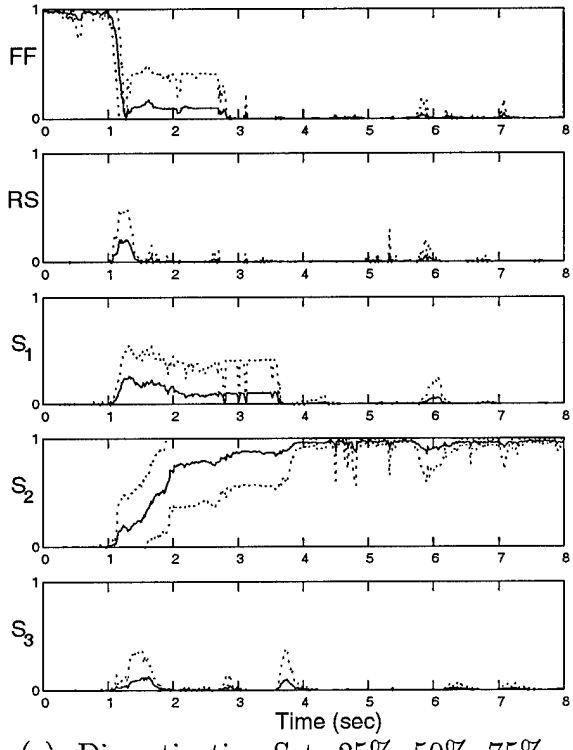
Figure 43. Probability Plot: Right Stabilator Failure, $\epsilon = 30\%$



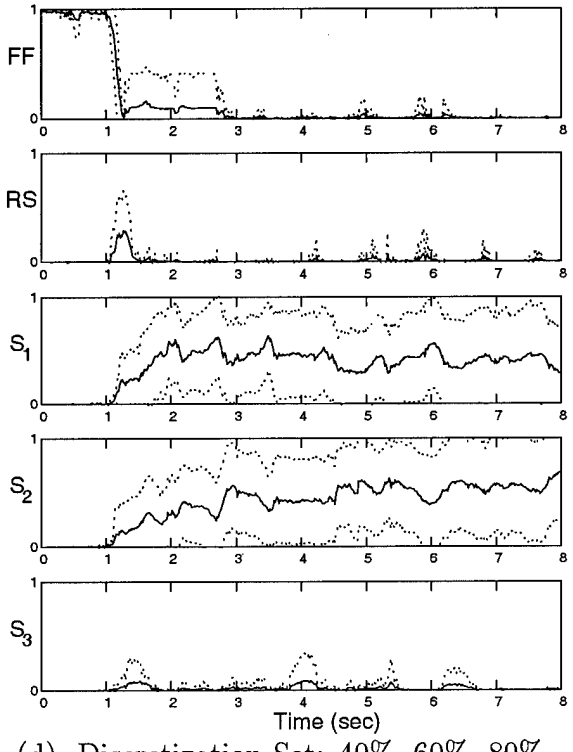
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

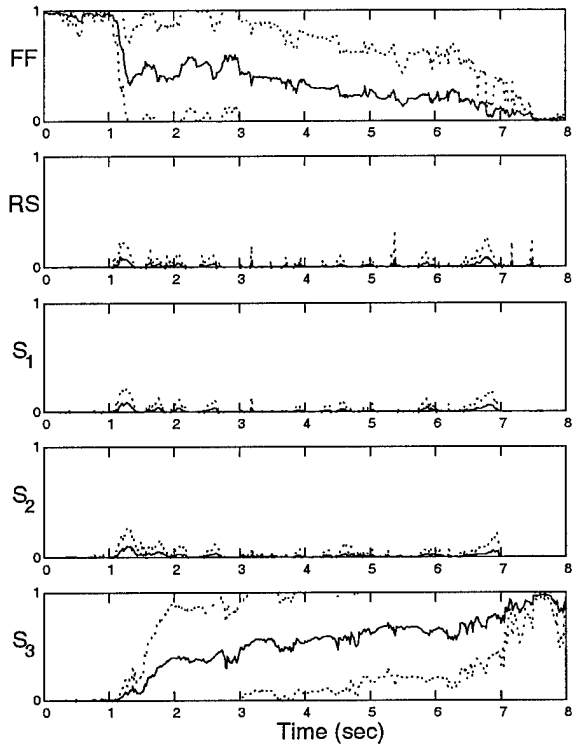


(c) Discretization Set: 25%, 50%, 75%

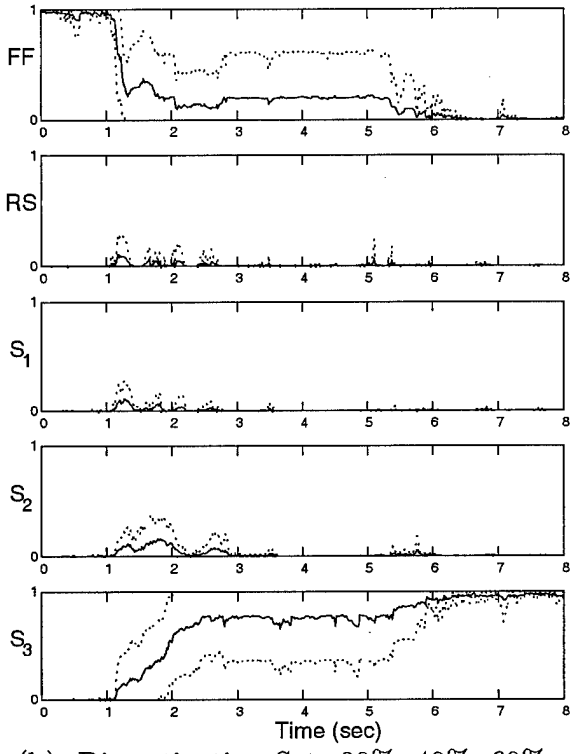


(d) Discretization Set: 40%, 60%, 80%

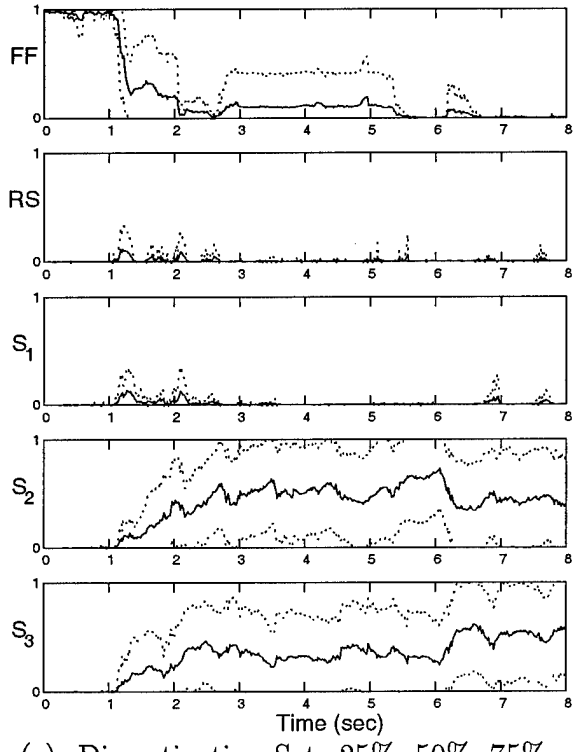
Figure 44. Probability Plot: Right Stabilator Failure, $\epsilon = 40\%$



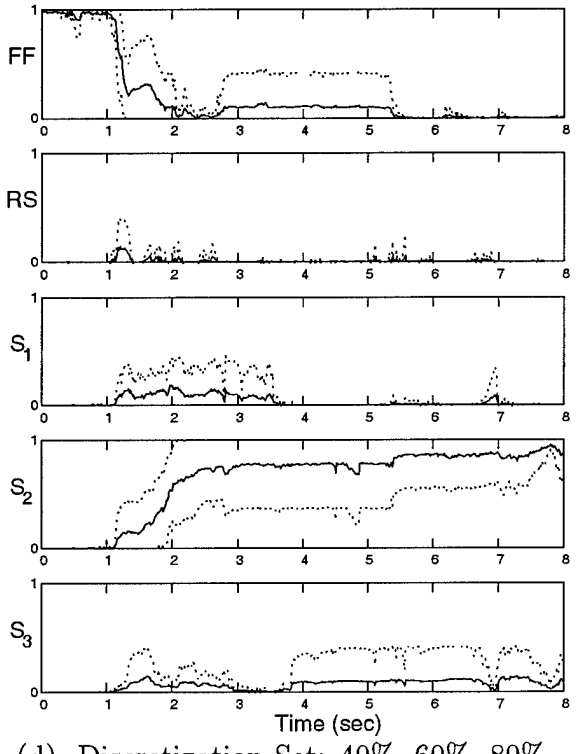
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

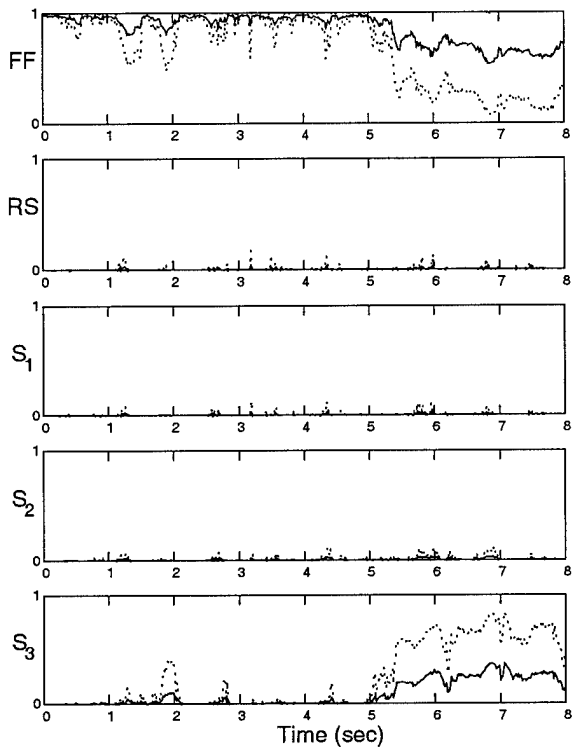


(c) Discretization Set: 25%, 50%, 75%

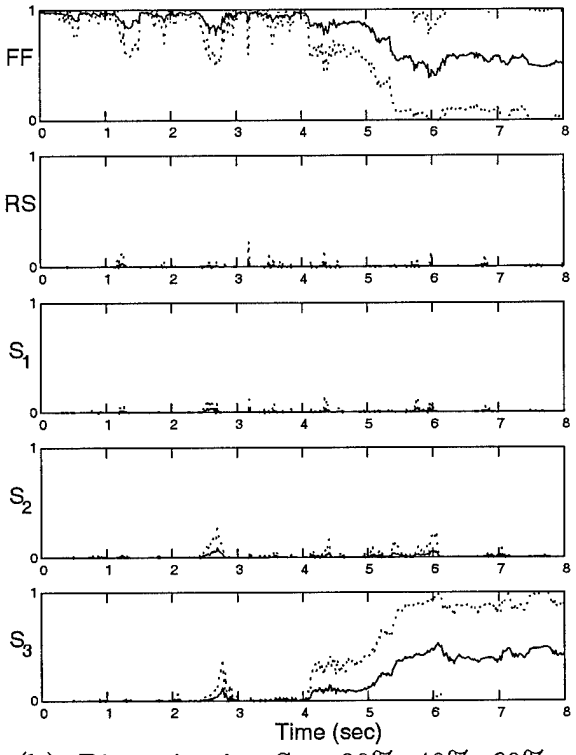


(d) Discretization Set: 40%, 60%, 80%

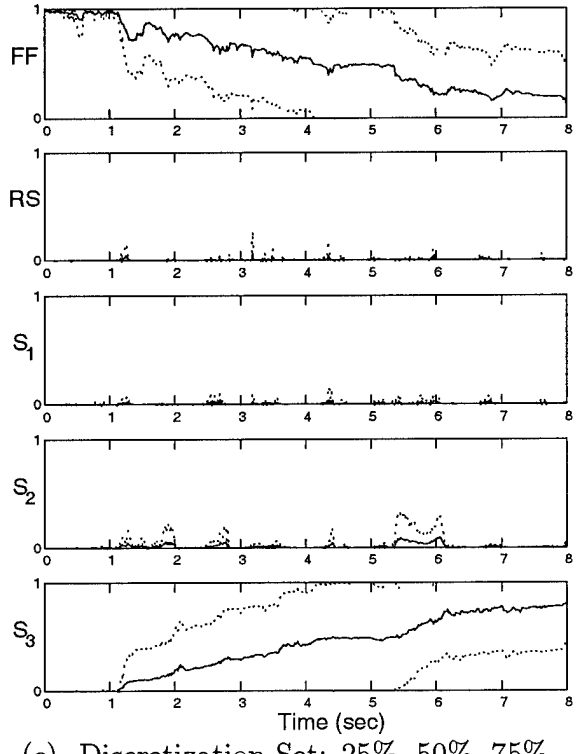
Figure 45. Probability Plot: Right Stabilator Failure, $\epsilon = 50\%$



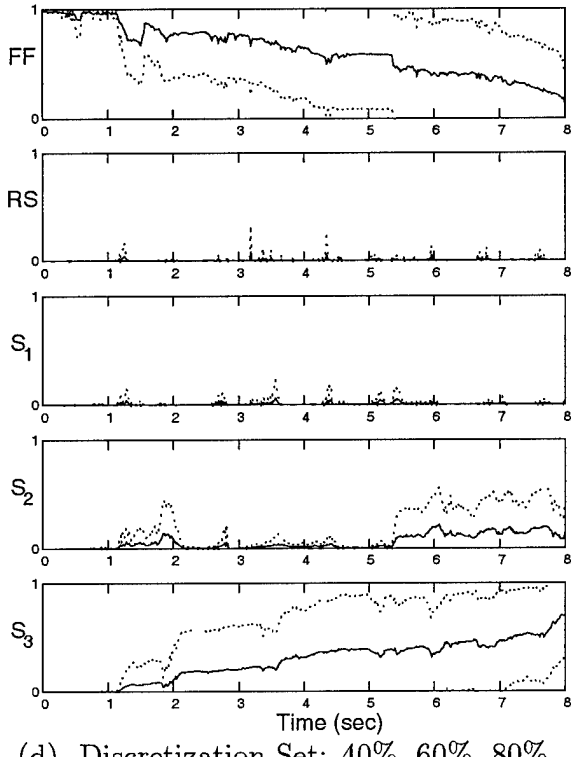
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

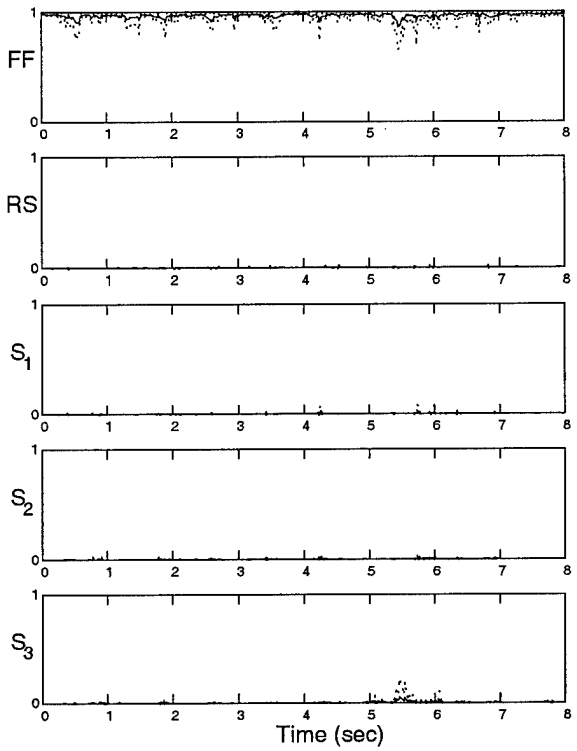


(c) Discretization Set: 25%, 50%, 75%

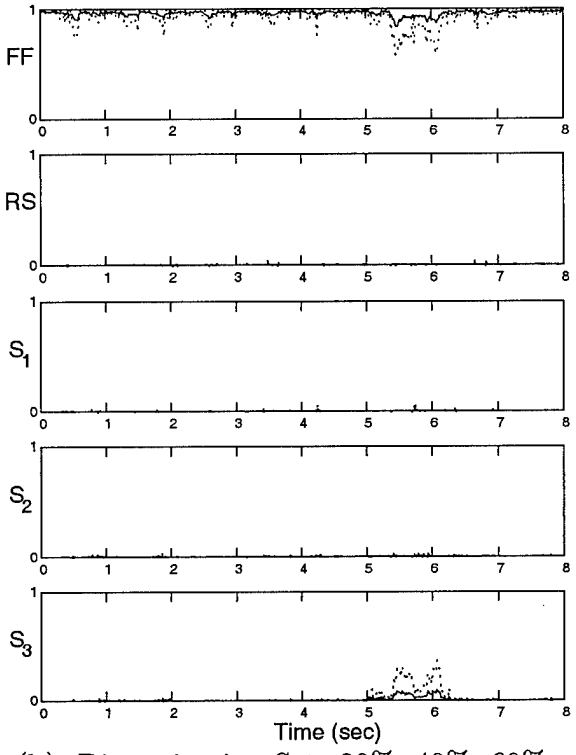


(d) Discretization Set: 40%, 60%, 80%

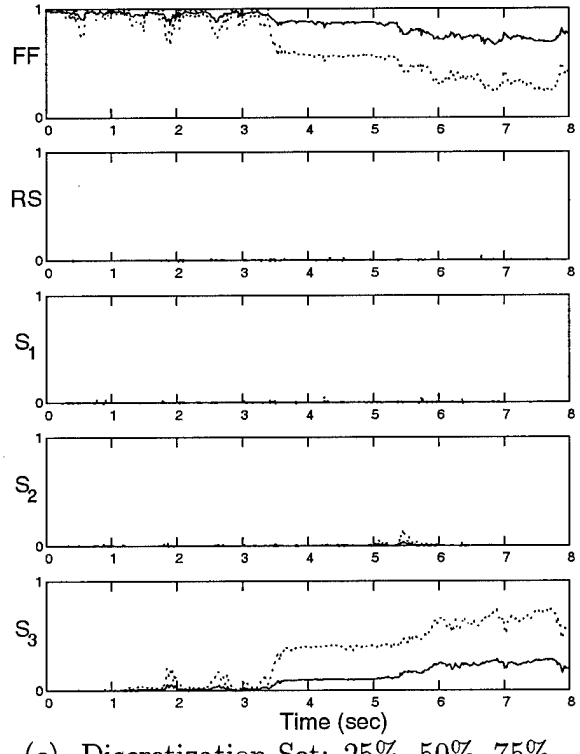
Figure 46. Probability Plot: Right Stabilator Failure, $\epsilon = 60\%$



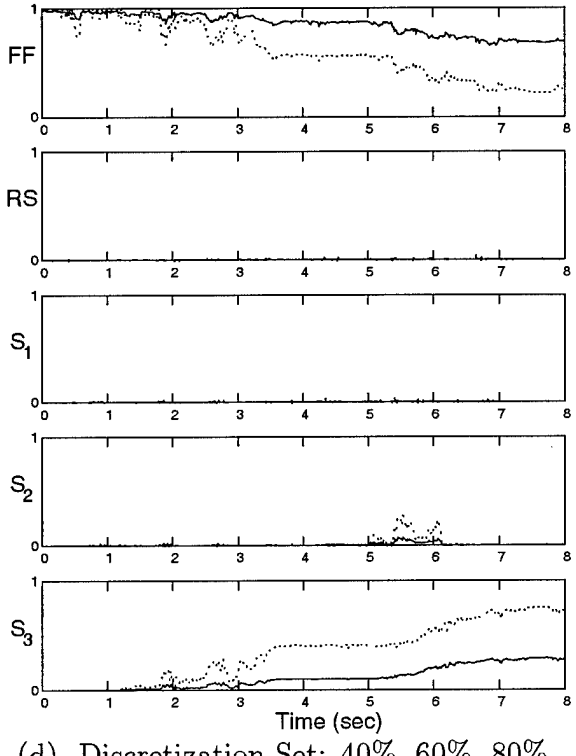
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

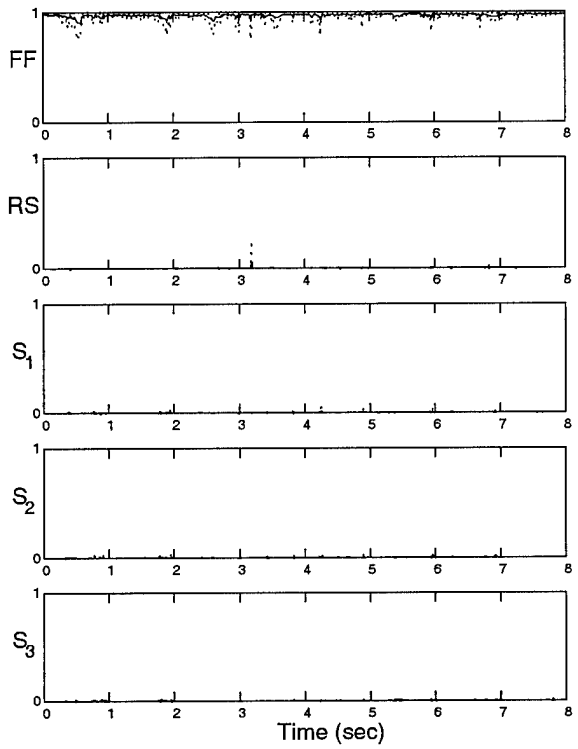


(c) Discretization Set: 25%, 50%, 75%

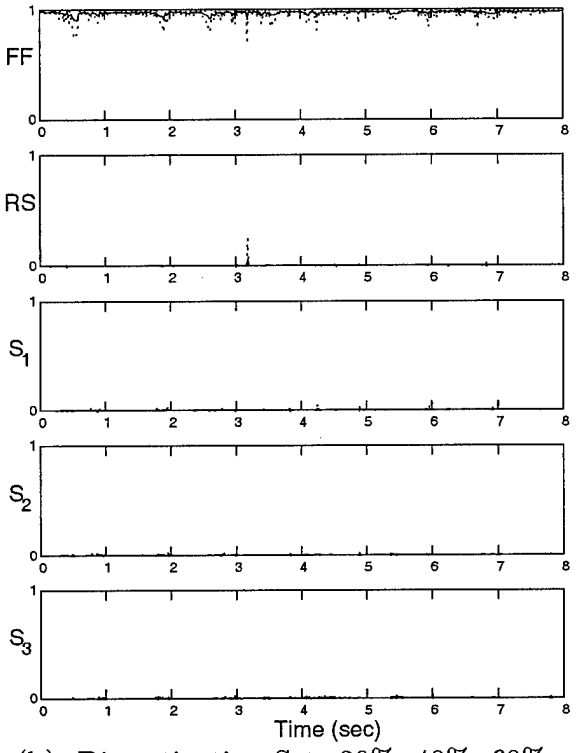


(d) Discretization Set: 40%, 60%, 80%

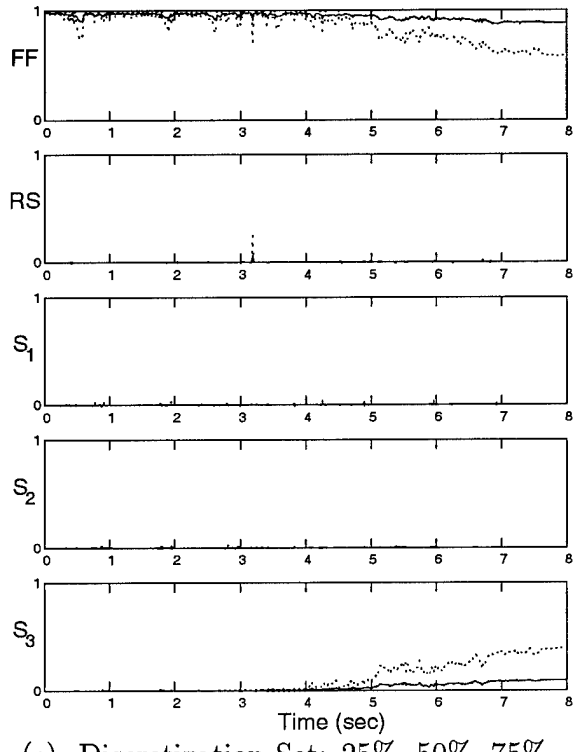
Figure 47. Probability Plot: Right Stabilator Failure, $\epsilon = 70\%$



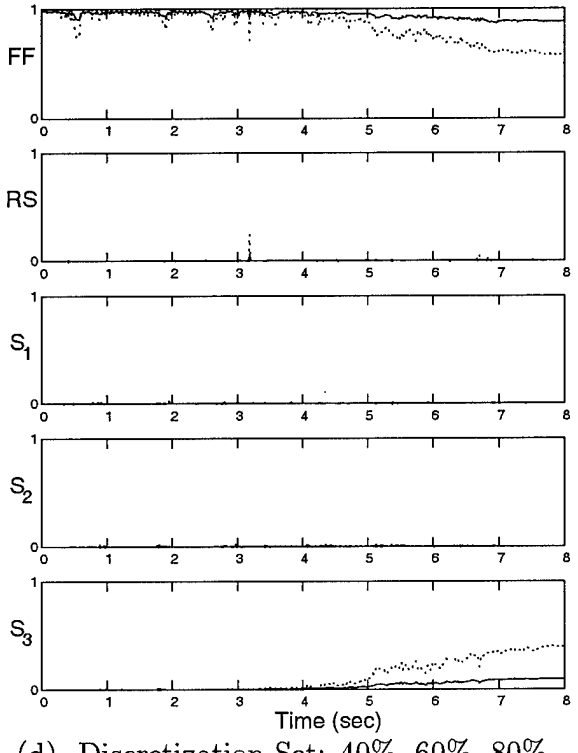
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

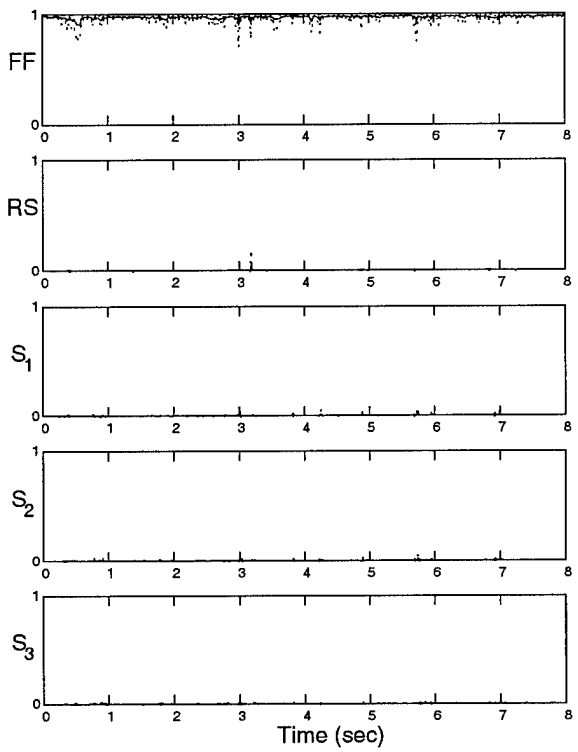


(c) Discretization Set: 25%, 50%, 75%

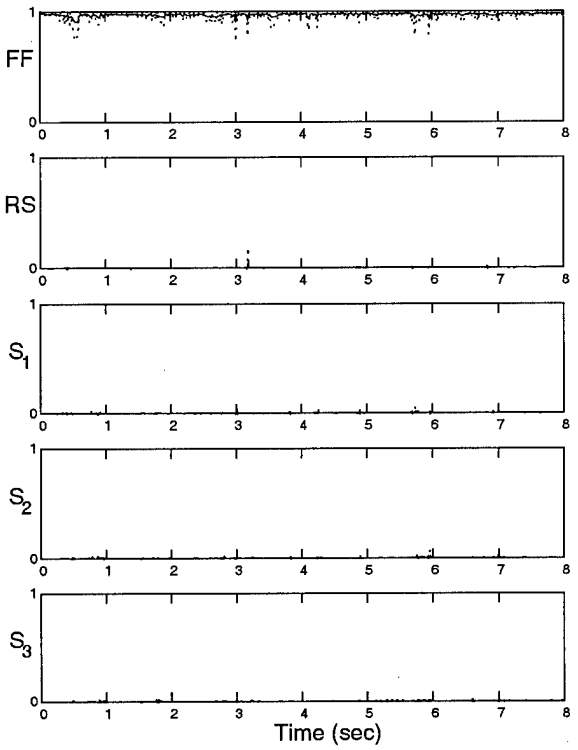


(d) Discretization Set: 40%, 60%, 80%

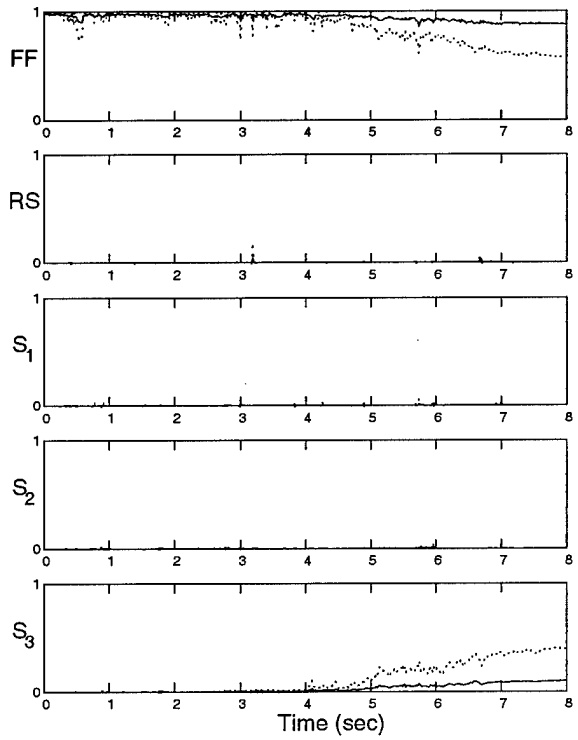
Figure 48. Probability Plot: Right Stabilator Failure, $\epsilon = 80\%$



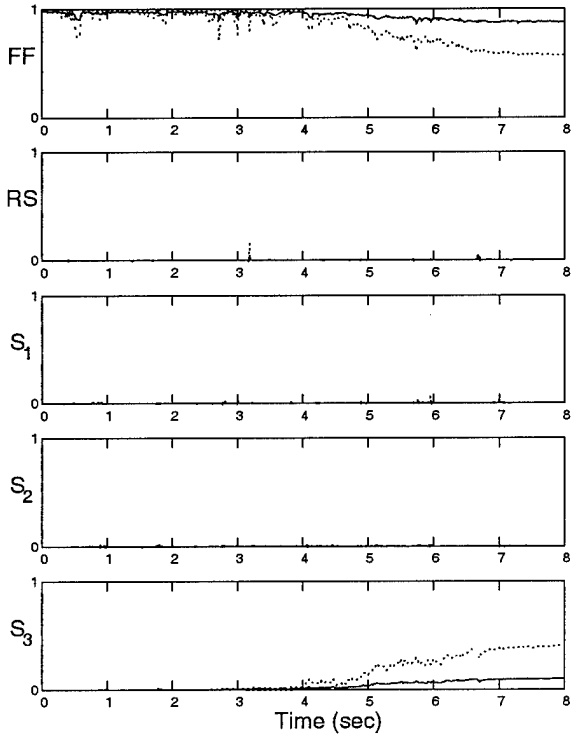
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

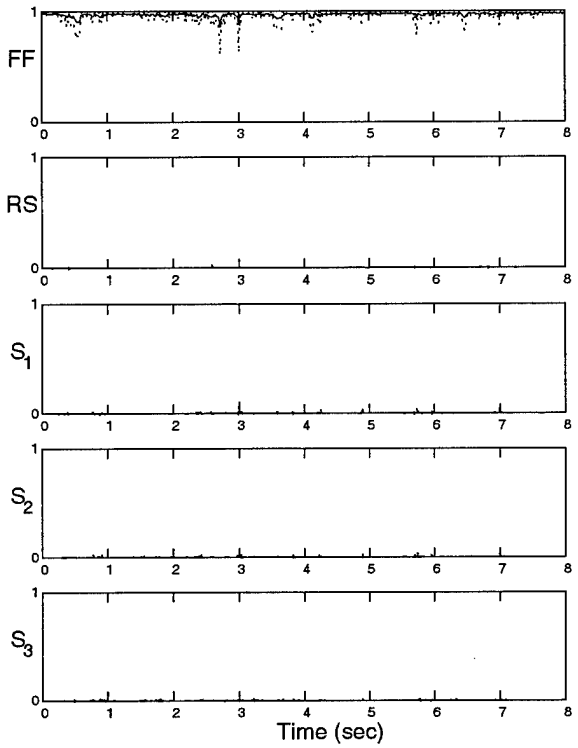


(c) Discretization Set: 25%, 50%, 75%

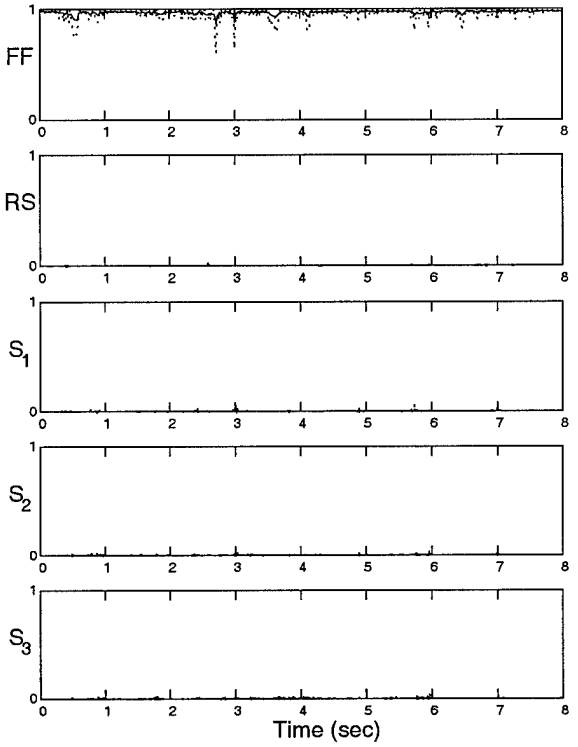


(d) Discretization Set: 40%, 60%, 80%

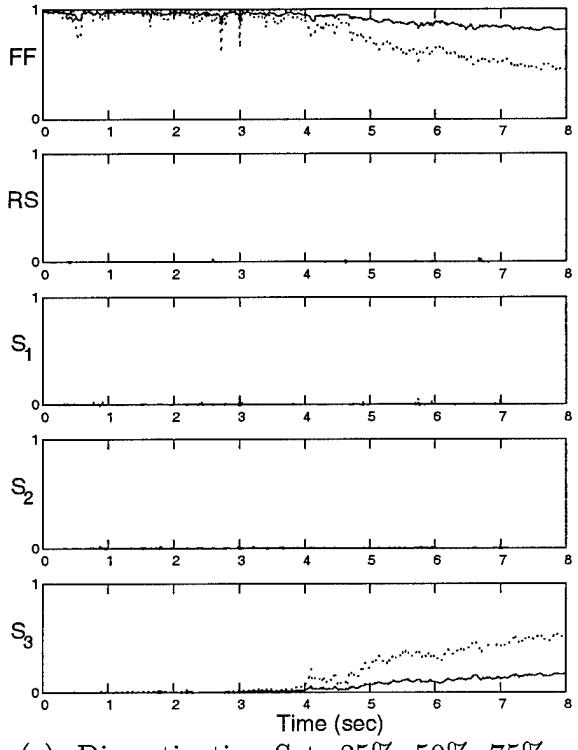
Figure 49. Probability Plot: Right Stabilator Failure, $\epsilon = 90\%$



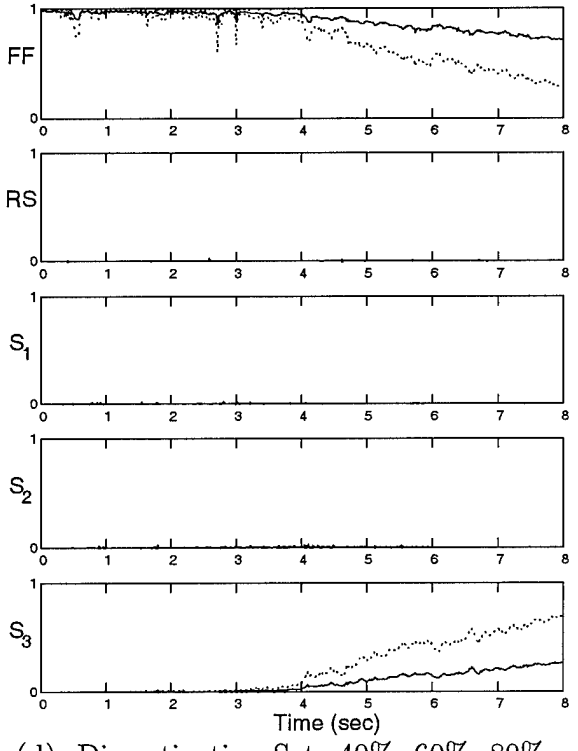
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



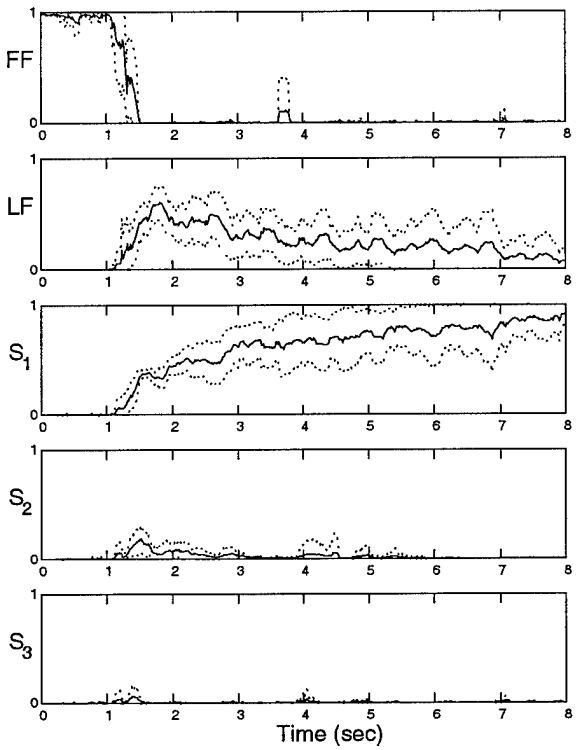
(d) Discretization Set: 40%, 60%, 80%

Figure 50. Probability Plot: Right Stabilator Failure, $\epsilon = 100\%$

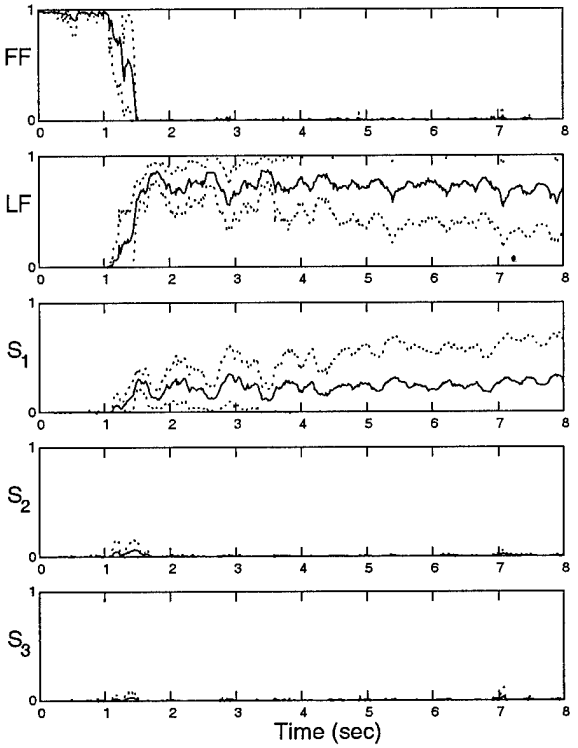
B.2 Left Flaperon

The corresponding figures for each effectiveness value of the left flaperon failure are given as

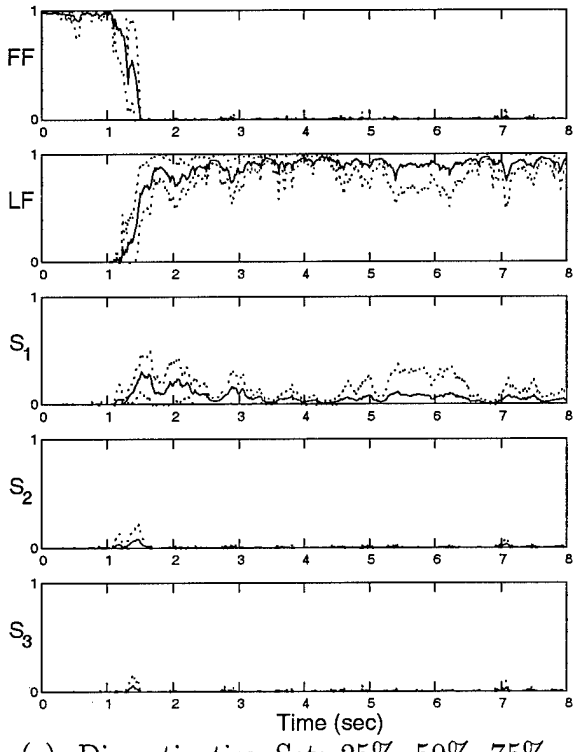
Figure	Left Flaperon Failure Effectiveness ϵ_{true}	Page
51	0%	148
52	10%	149
53	20%	150
54	30%	151
55	40%	152
56	50%	153
57	60%	154
58	70%	155
59	80%	156
60	90%	157
61	100%	158



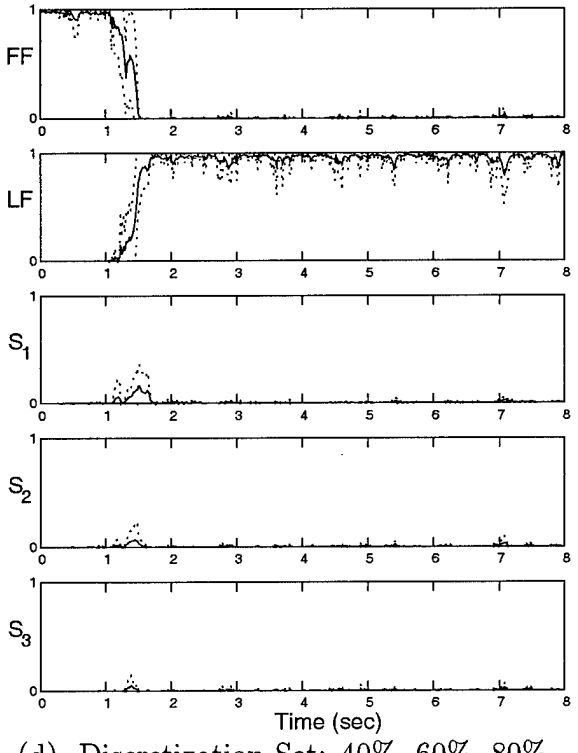
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

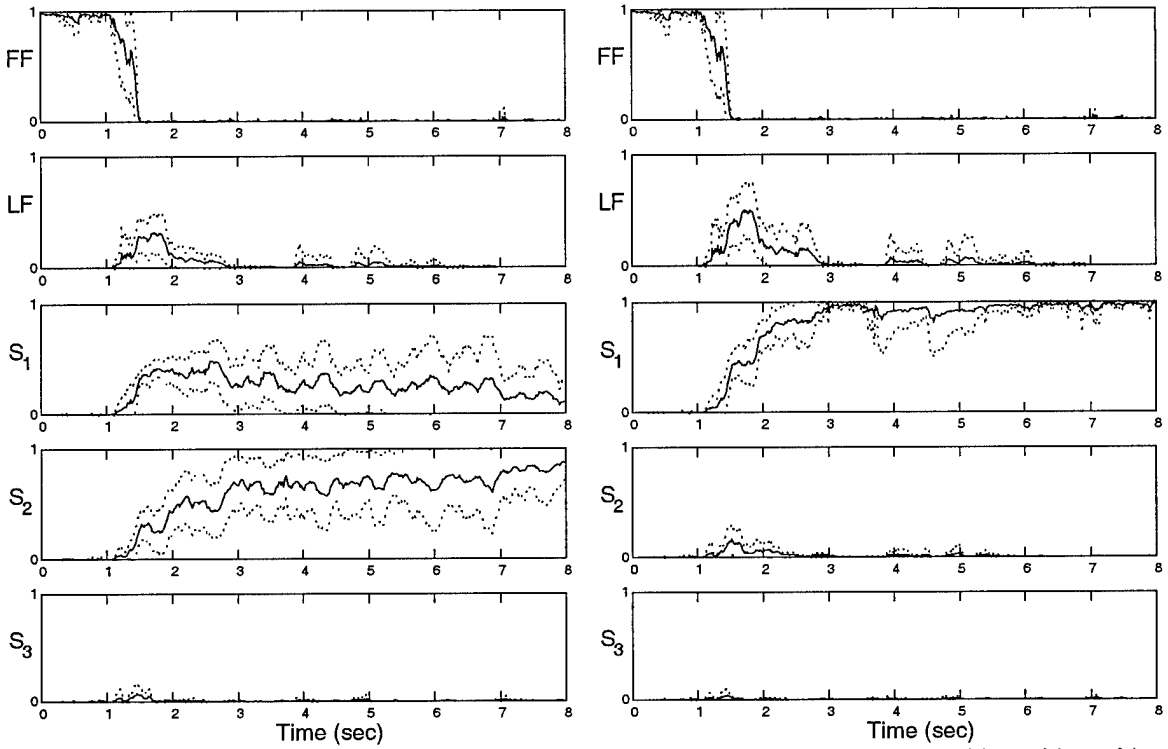


(c) Discretization Set: 25%, 50%, 75%



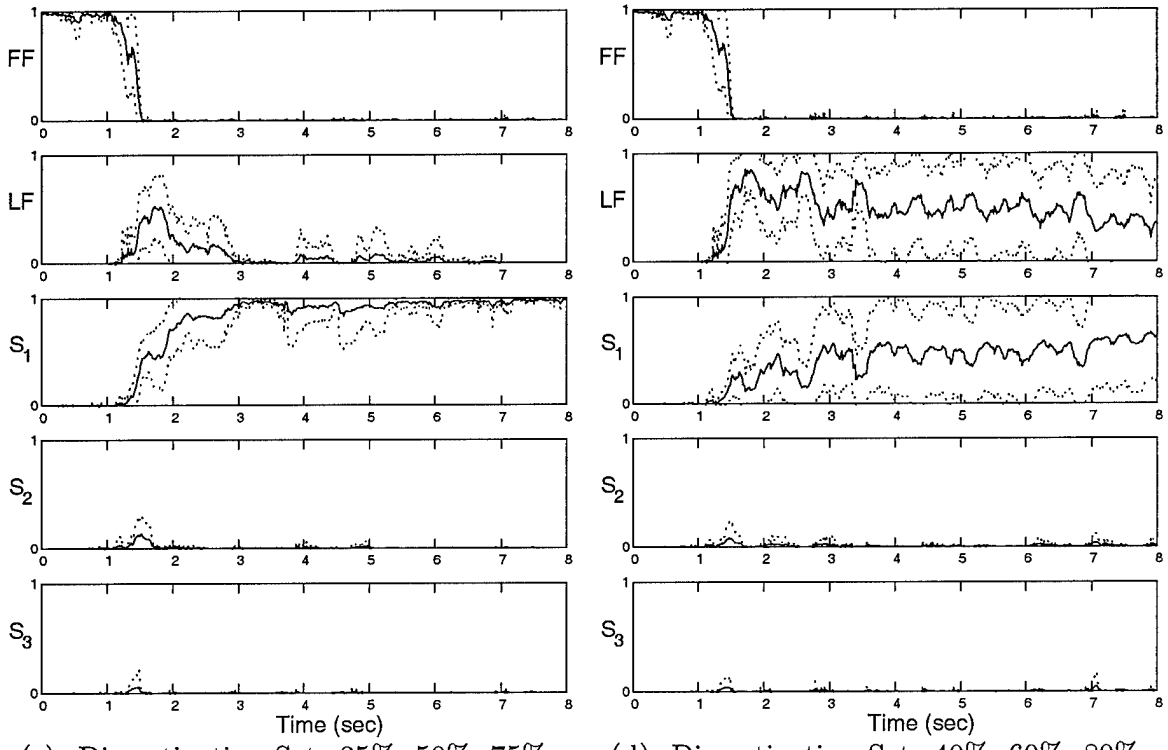
(d) Discretization Set: 40%, 60%, 80%

Figure 51. Probability Plot: Left Flaperon Failure, $\epsilon = 0\%$



(a) Discretization Set: 10%, 25%, 50%

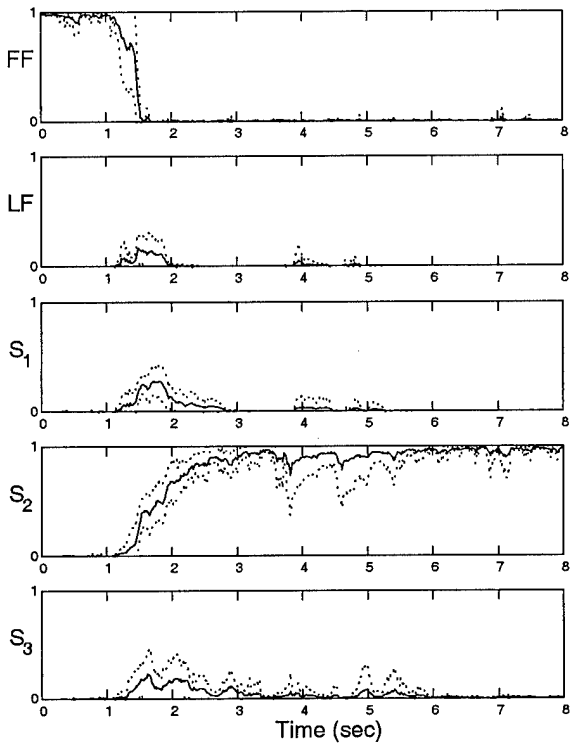
(b) Discretization Set: 20%, 40%, 60%



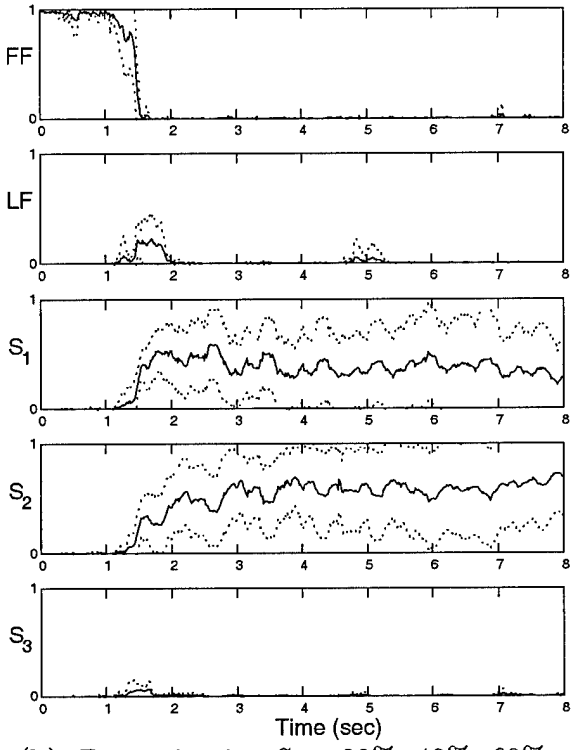
(c) Discretization Set: 25%, 50%, 75%

(d) Discretization Set: 40%, 60%, 80%

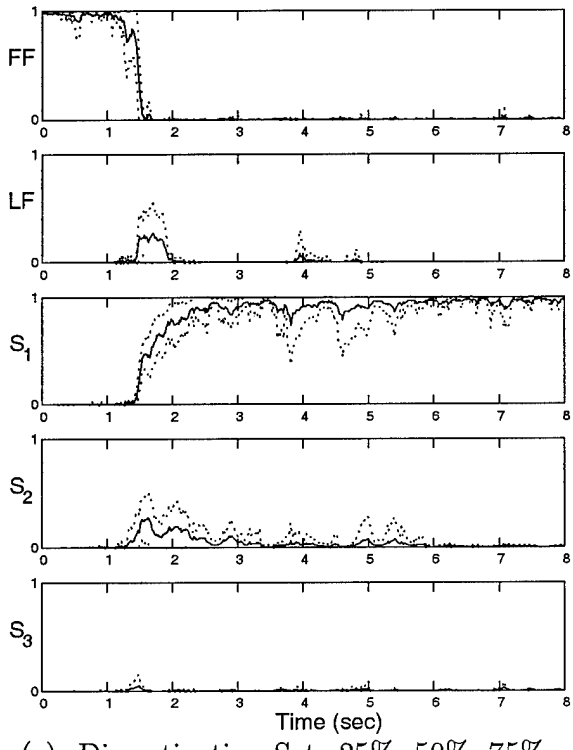
Figure 52. Probability Plot: Left Flaperon Failure, $\epsilon = 10\%$



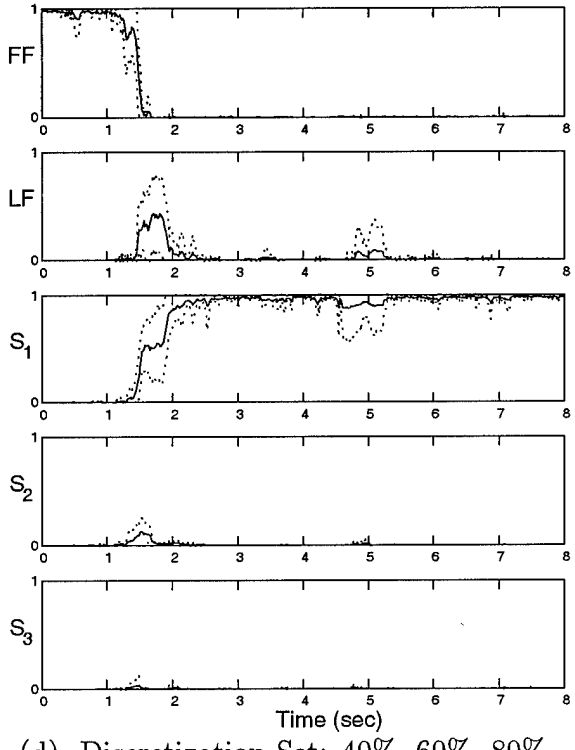
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

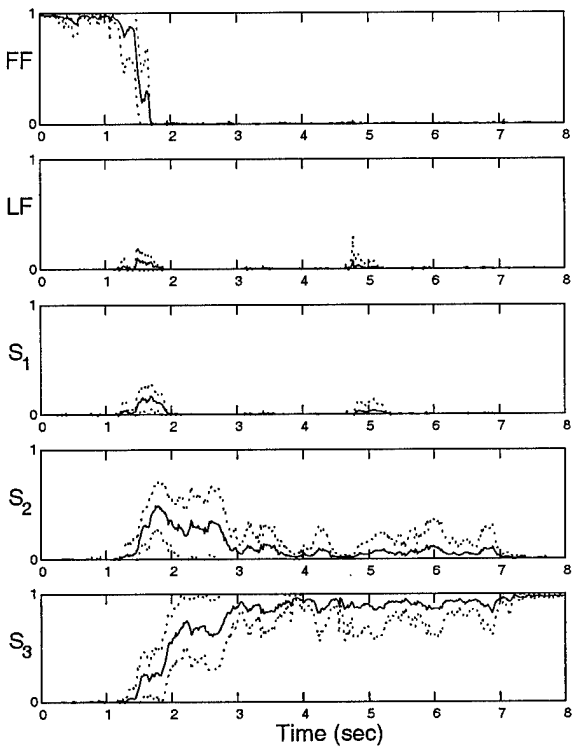


(c) Discretization Set: 25%, 50%, 75%

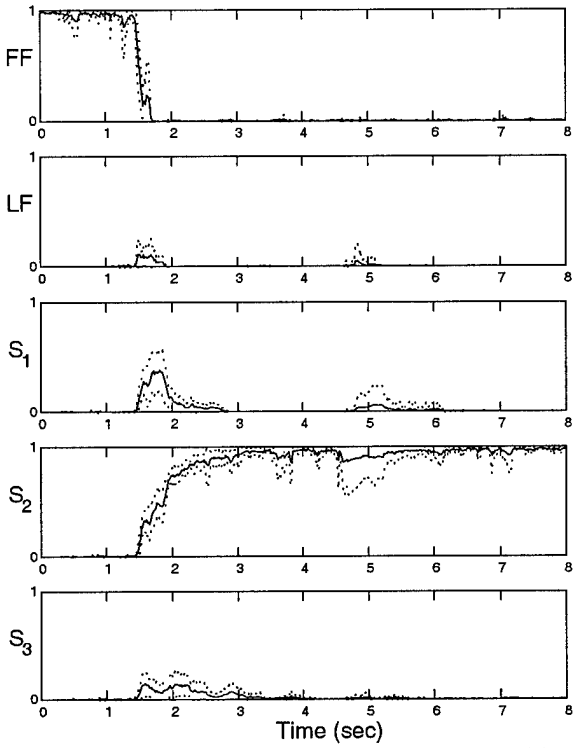


(d) Discretization Set: 40%, 60%, 80%

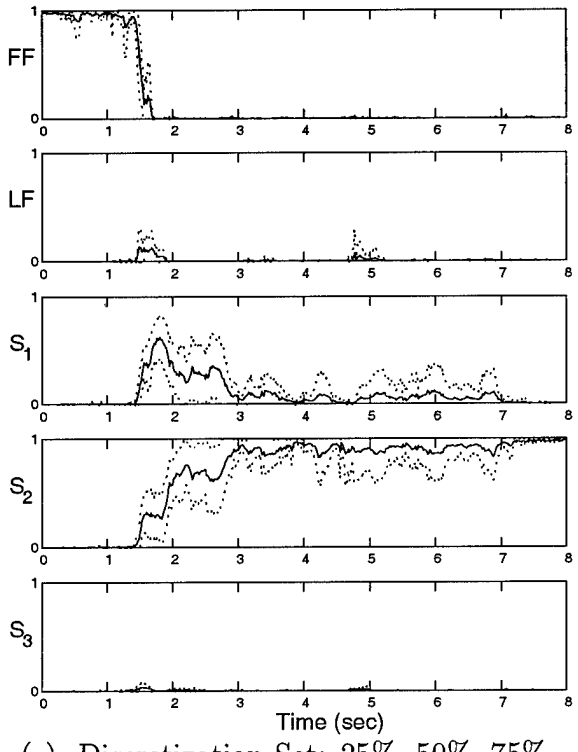
Figure 53. Probability Plot: Left Flaperon Failure, $\epsilon = 20\%$



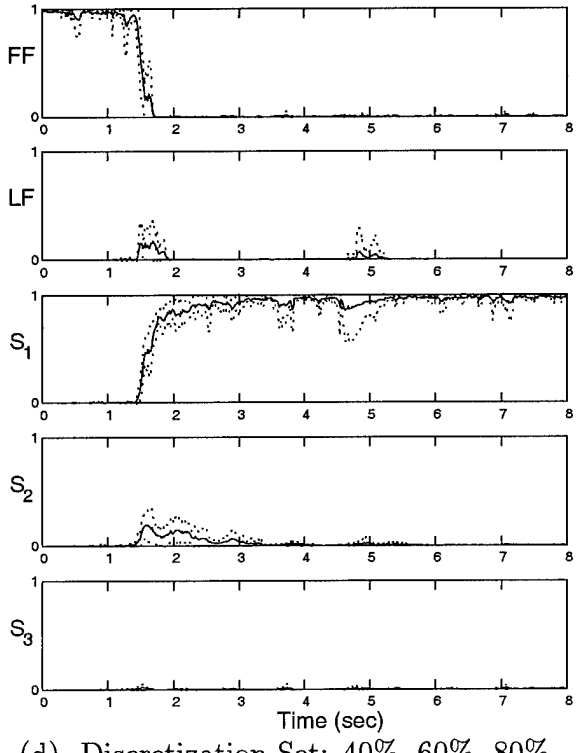
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

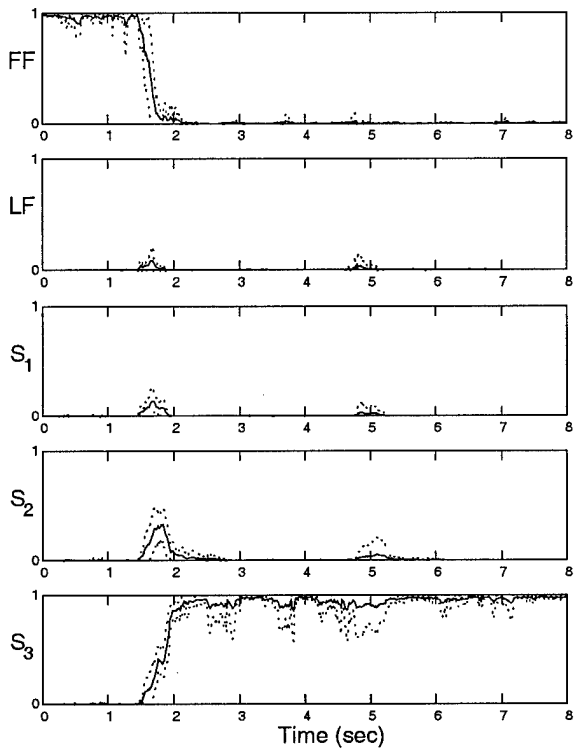


(c) Discretization Set: 25%, 50%, 75%

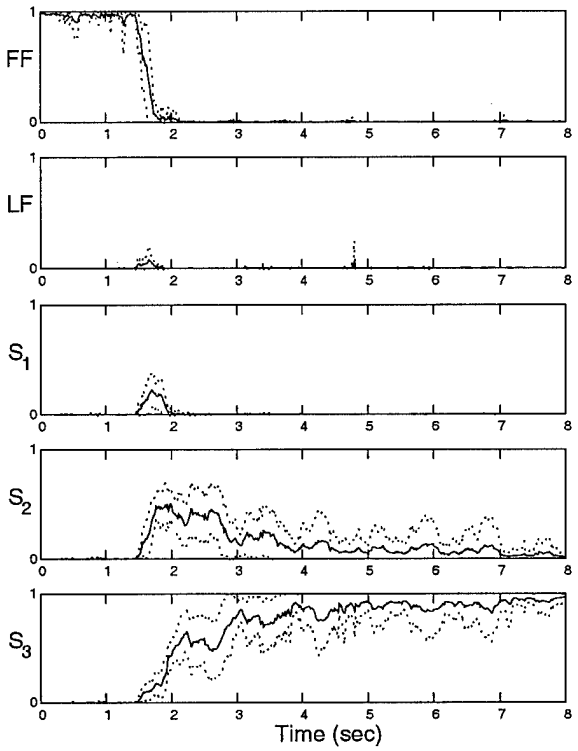


(d) Discretization Set: 40%, 60%, 80%

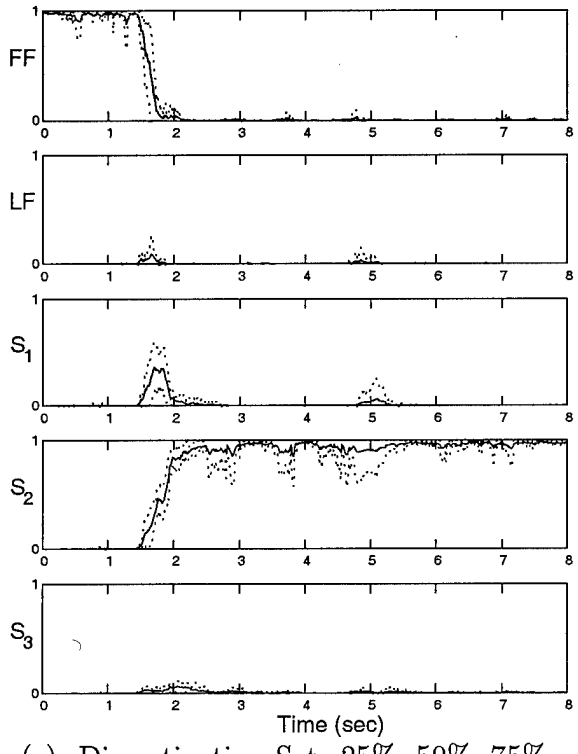
Figure 54. Probability Plot: Left Flaperon Failure, $\epsilon = 30\%$



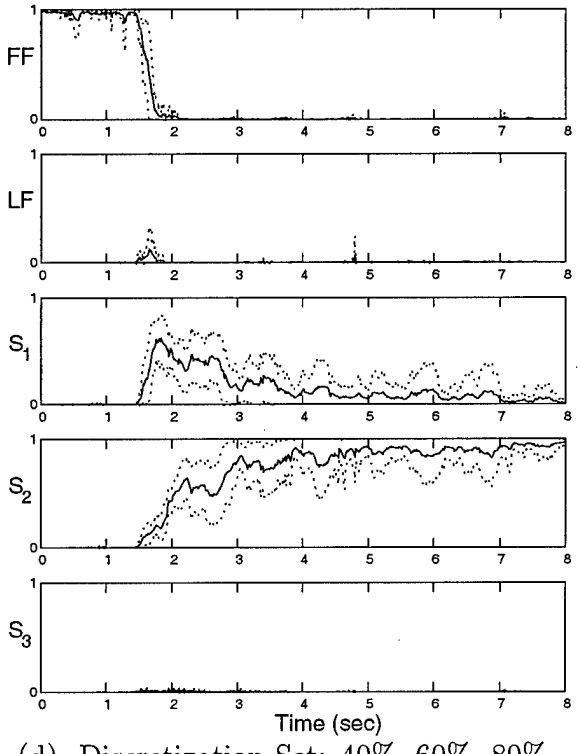
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

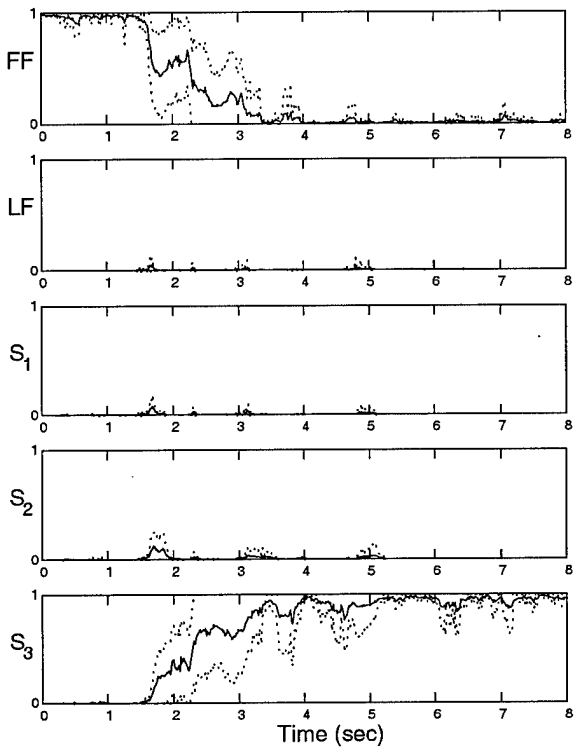


(c) Discretization Set: 25%, 50%, 75%

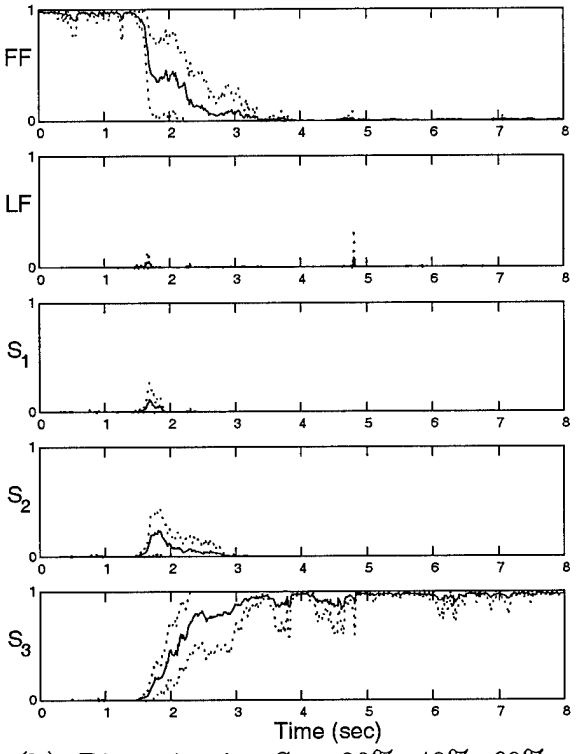


(d) Discretization Set: 40%, 60%, 80%

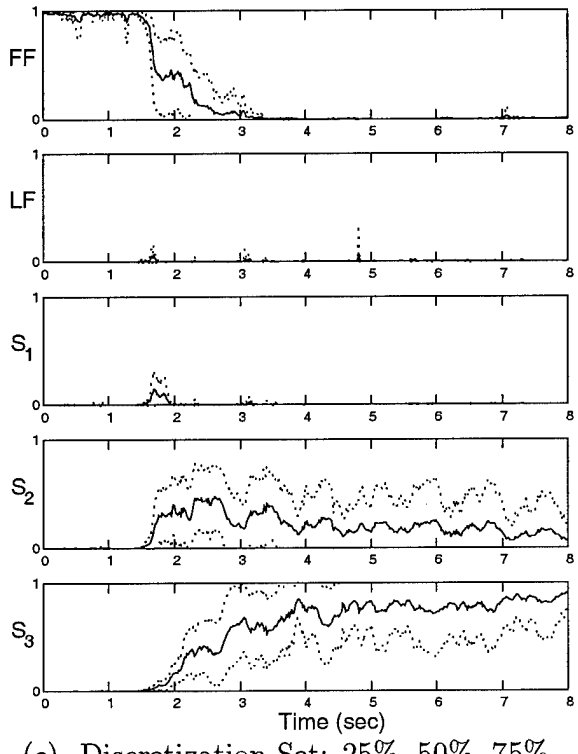
Figure 55. Probability Plot: Left Flaperon Failure, $\epsilon = 40\%$



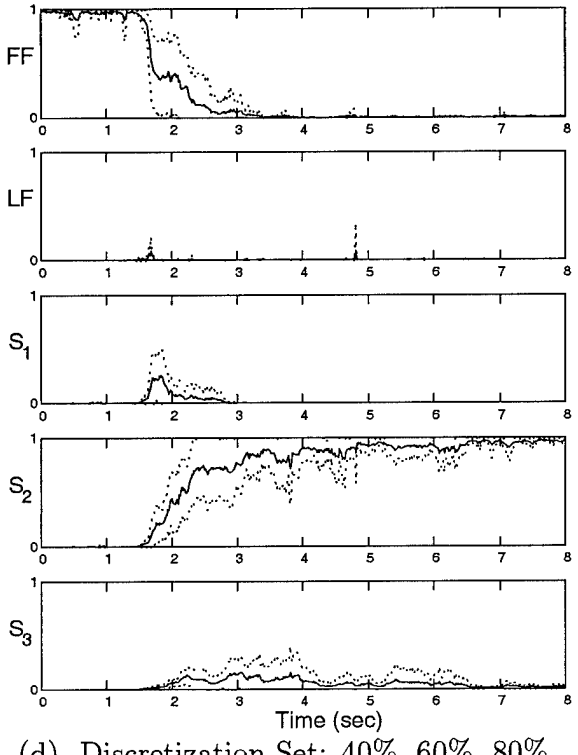
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

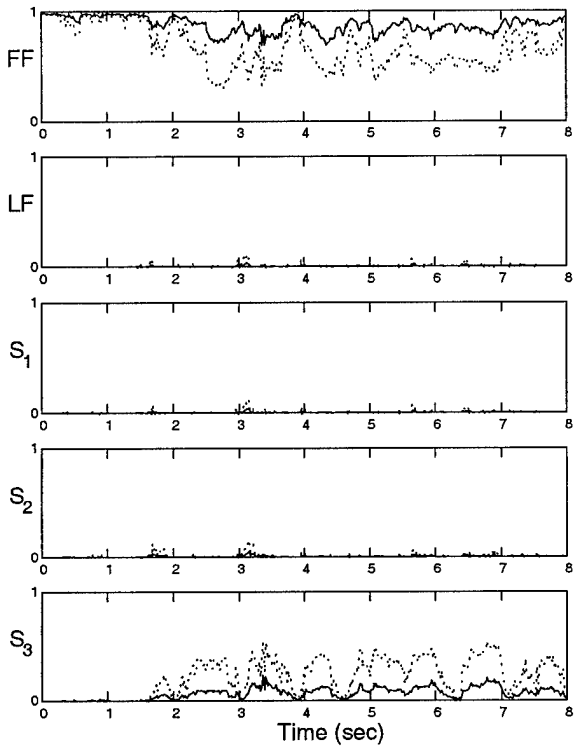


(c) Discretization Set: 25%, 50%, 75%

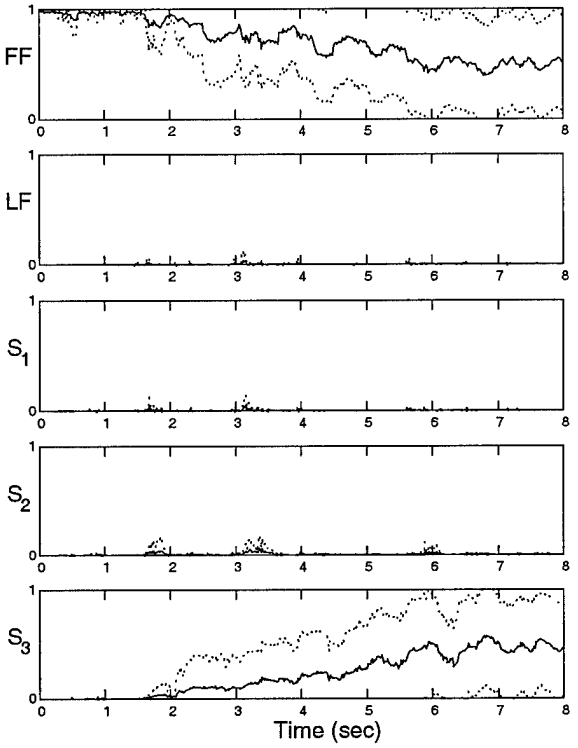


(d) Discretization Set: 40%, 60%, 80%

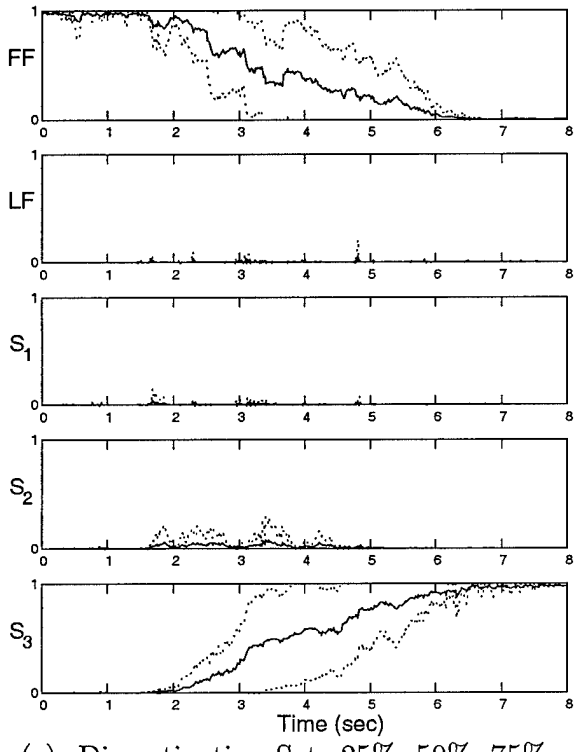
Figure 56. Probability Plot: Left Flaperon Failure, $\epsilon = 50\%$



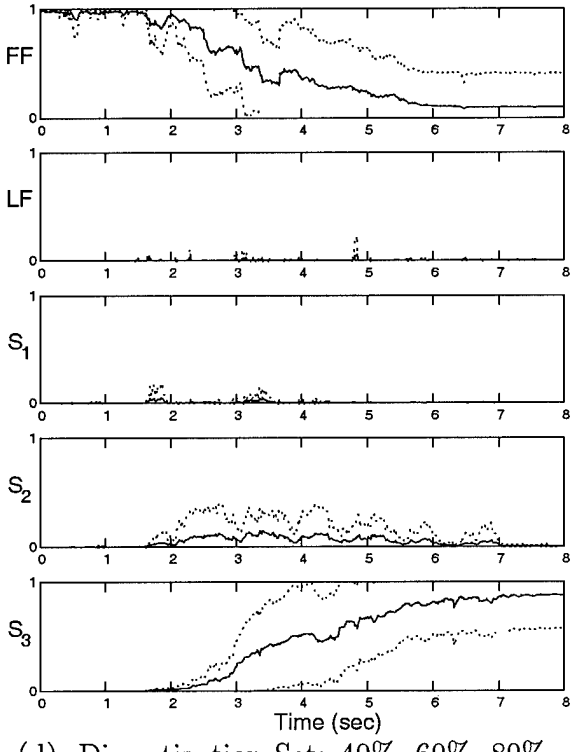
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

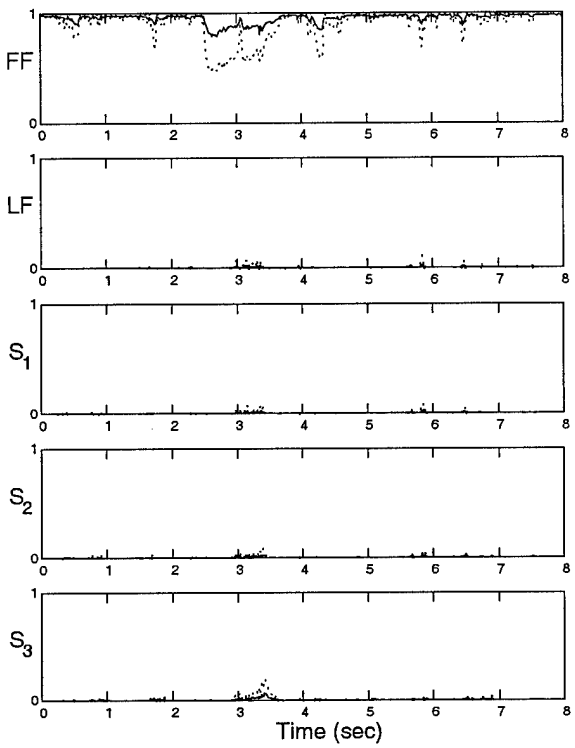


(c) Discretization Set: 25%, 50%, 75%

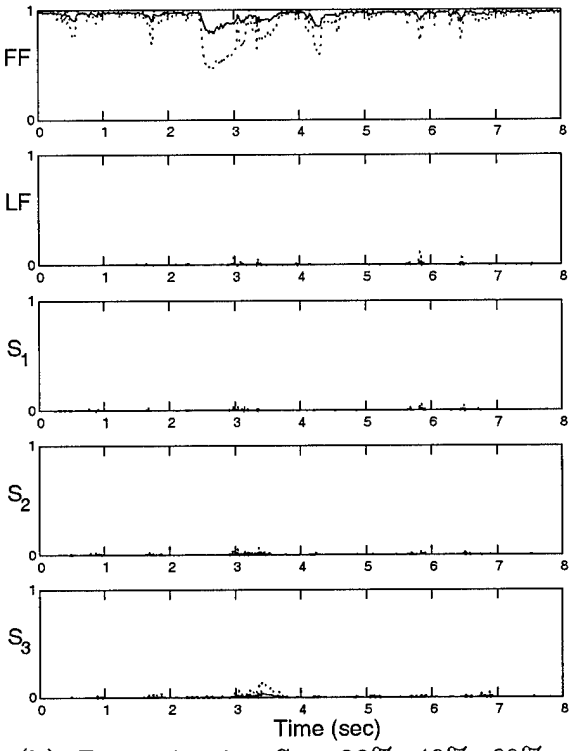


(d) Discretization Set: 40%, 60%, 80%

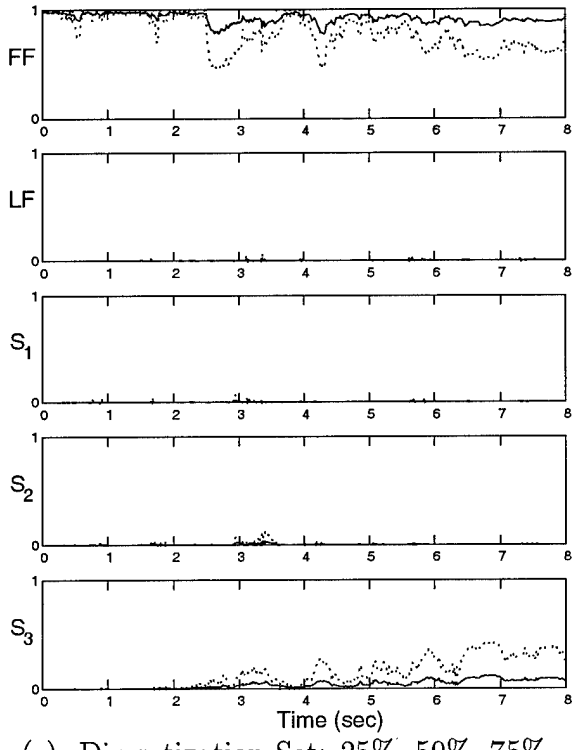
Figure 57. Probability Plot: Left Flaperon Failure, $\epsilon = 60\%$



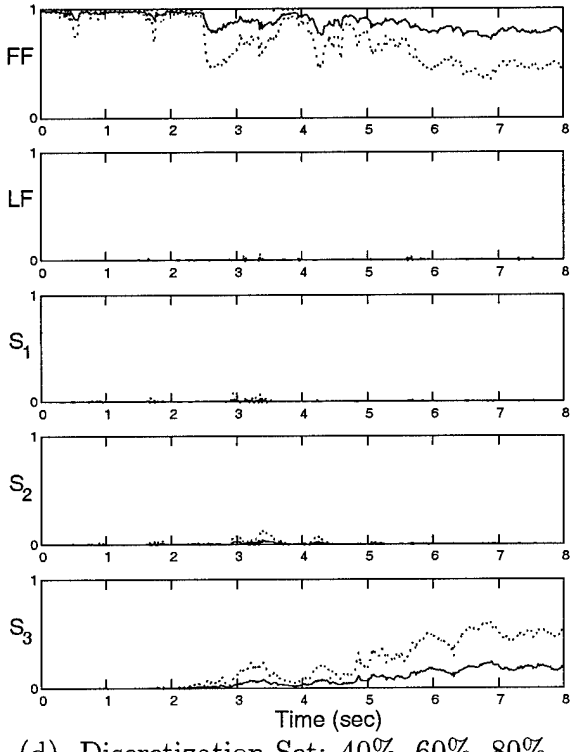
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

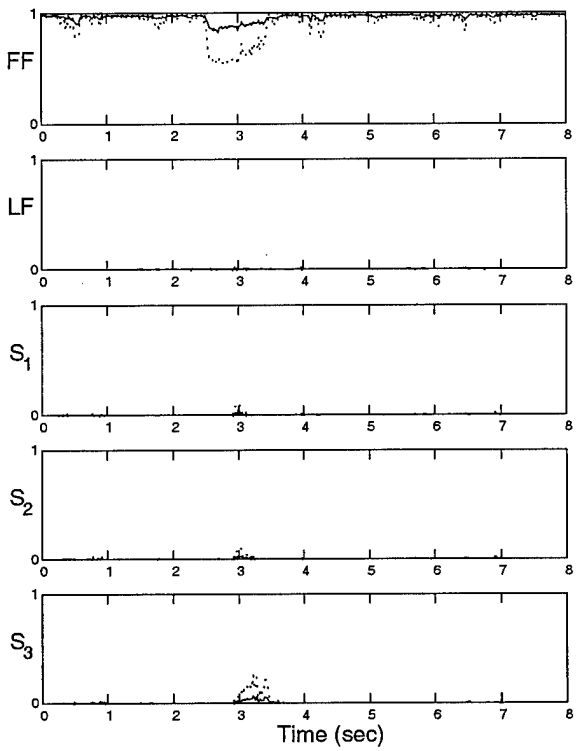


(c) Discretization Set: 25%, 50%, 75%

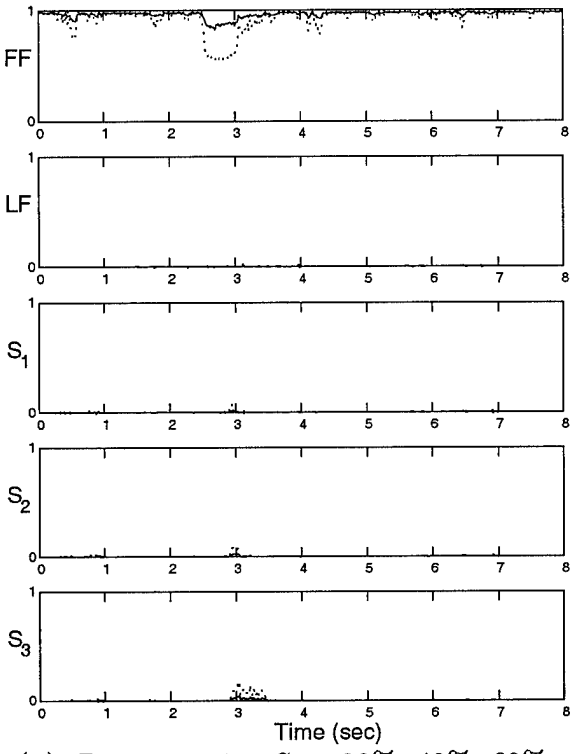


(d) Discretization Set: 40%, 60%, 80%

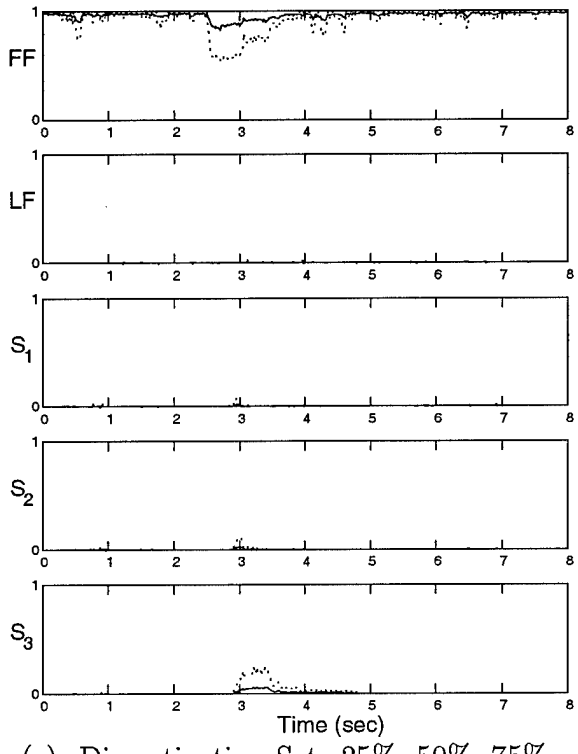
Figure 58. Probability Plot: Left Flaperon Failure, $\epsilon = 70\%$



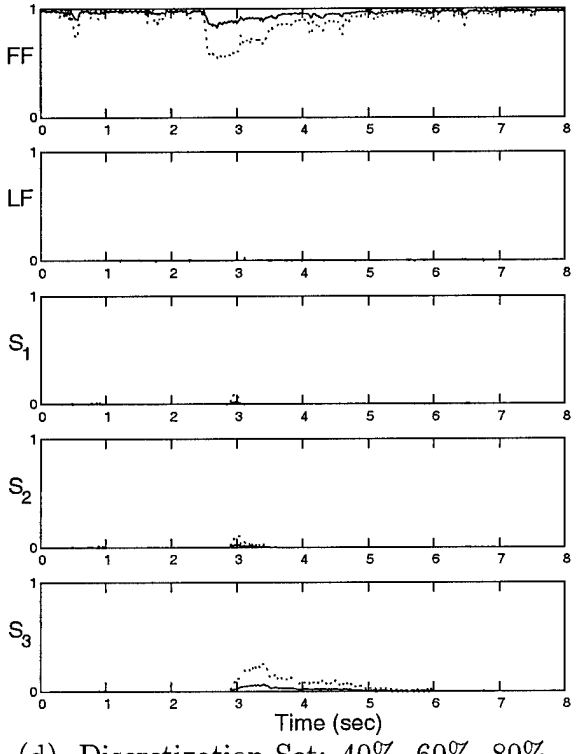
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

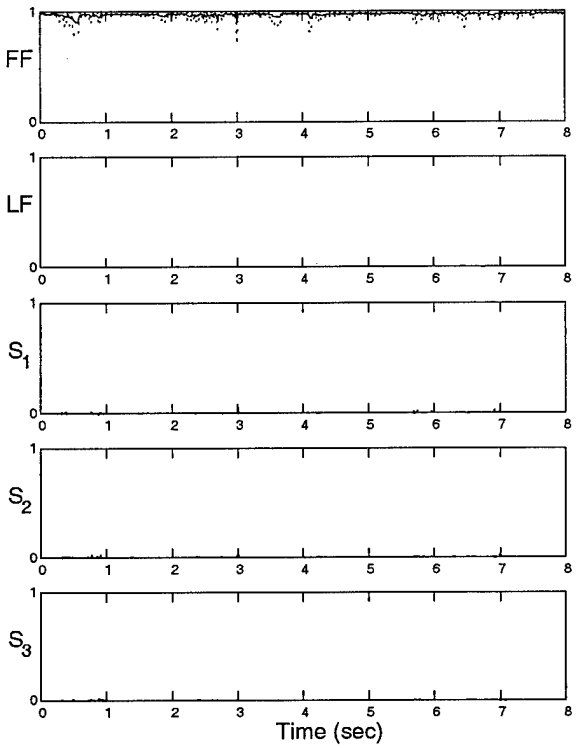


(c) Discretization Set: 25%, 50%, 75%

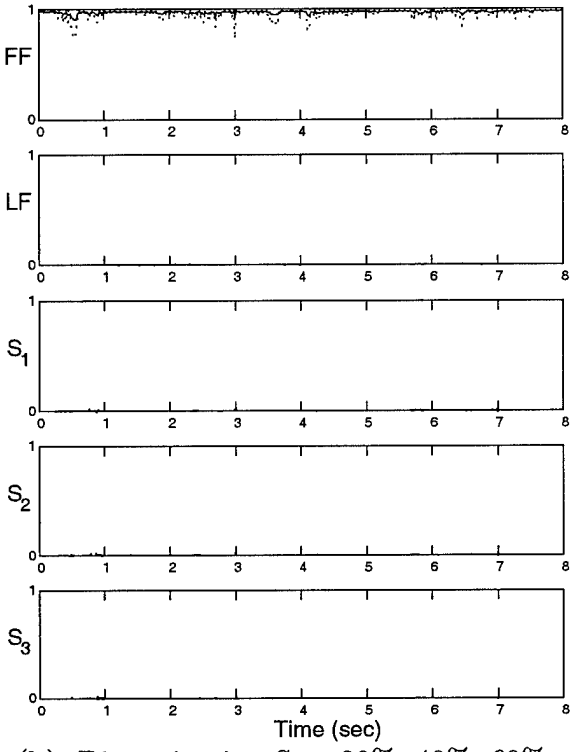


(d) Discretization Set: 40%, 60%, 80%

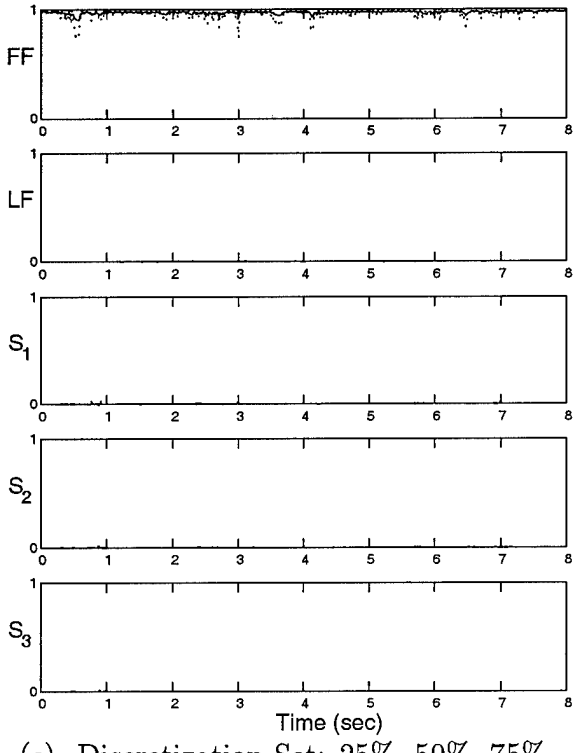
Figure 59. Probability Plot: Left Flaperon Failure, $\epsilon = 80\%$



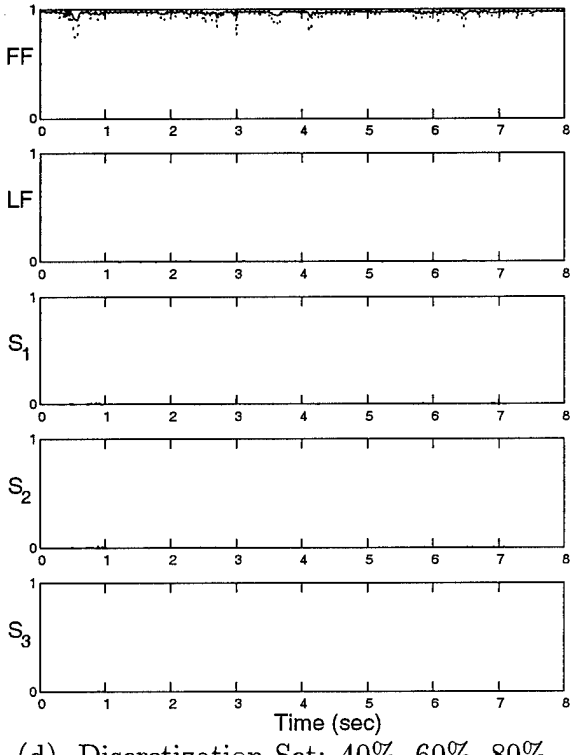
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

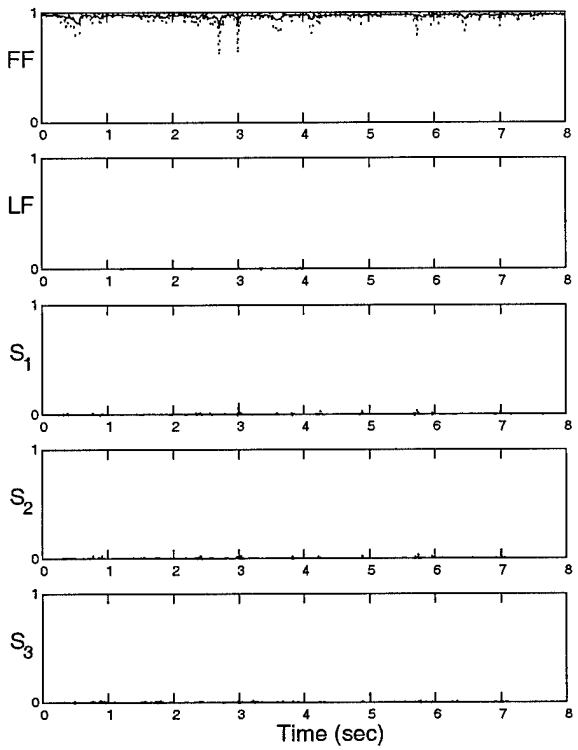


(c) Discretization Set: 25%, 50%, 75%

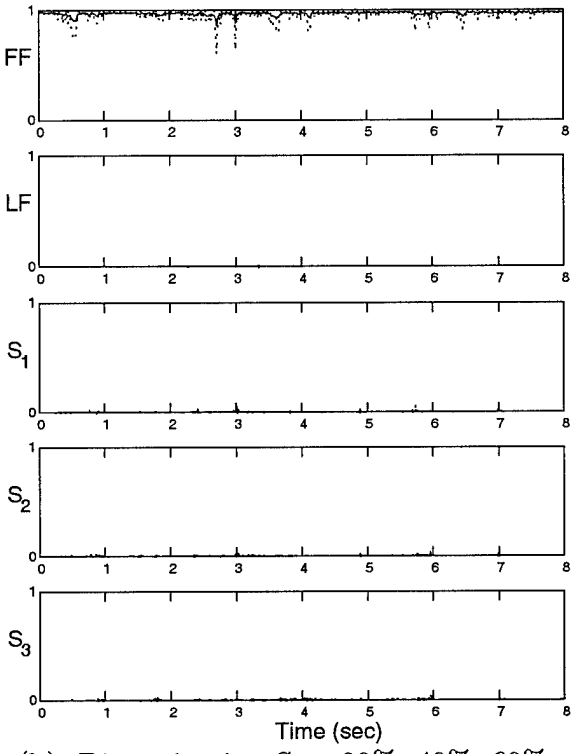


(d) Discretization Set: 40%, 60%, 80%

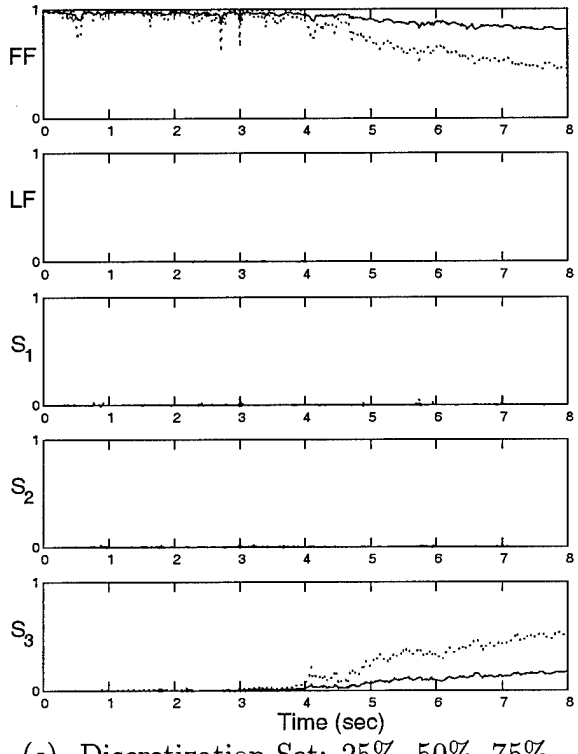
Figure 60. Probability Plot: Left Flaperon Failure, $\epsilon = 90\%$



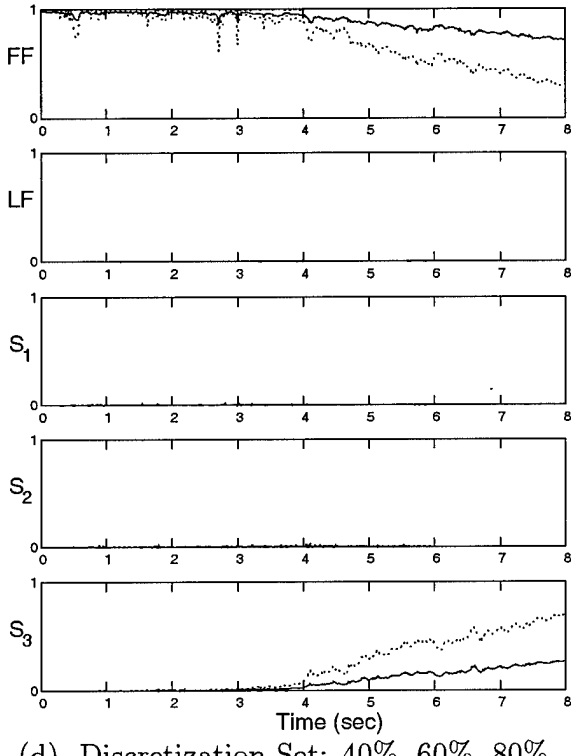
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



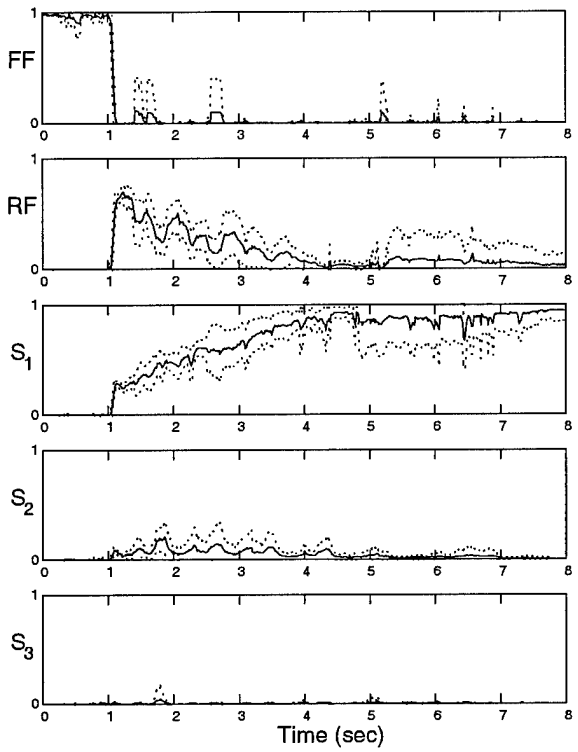
(d) Discretization Set: 40%, 60%, 80%

Figure 61. Probability Plot: Left Flaperon Failure, $\epsilon = 100\%$

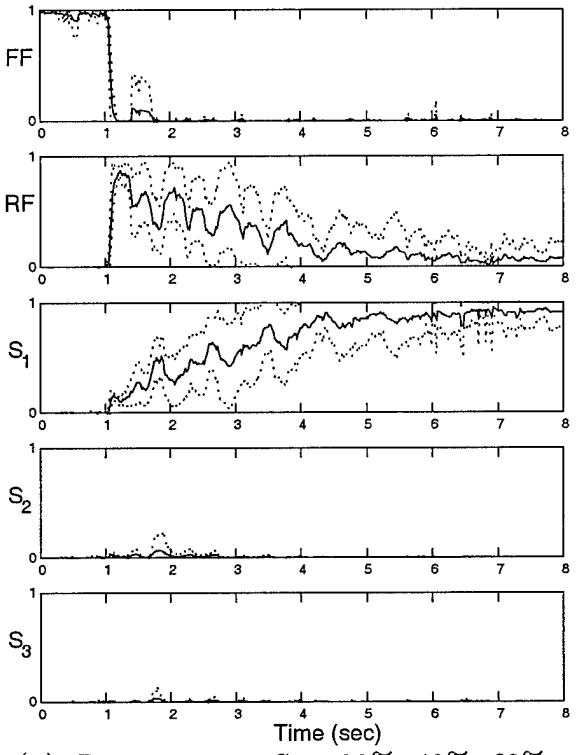
B.3 Right Flaperon

The corresponding figures for each effectiveness value of the right flaperon failure are given as

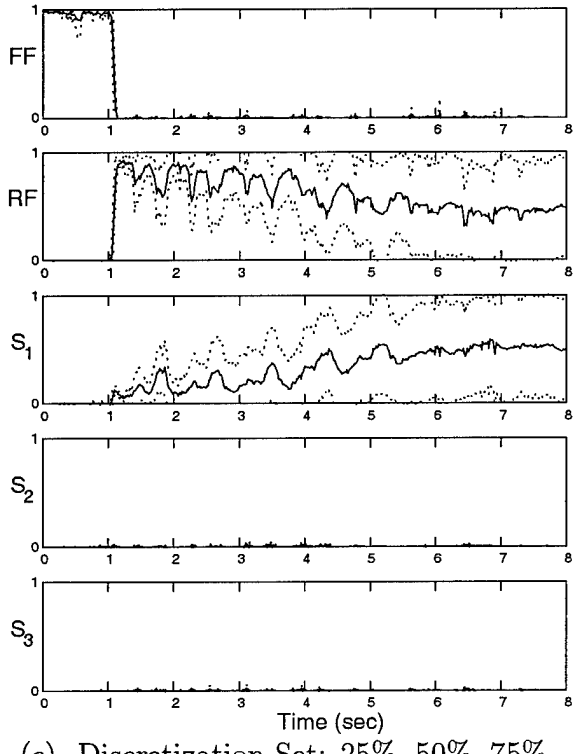
Figure	Right Flaperon Failure Effectiveness ϵ_{true}	Page
62	0%	160
63	10%	161
64	20%	162
65	30%	163
66	40%	164
67	50%	165
68	60%	166
69	70%	167
70	80%	168
71	90%	169
72	100%	170



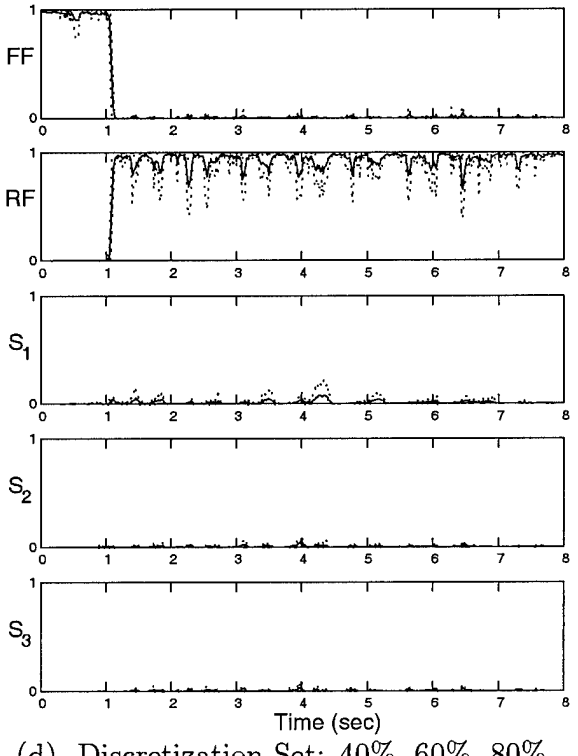
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

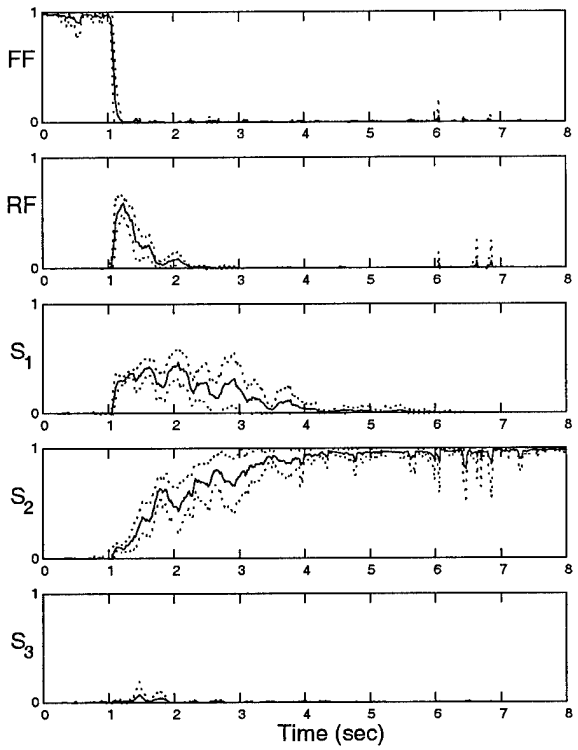


(c) Discretization Set: 25%, 50%, 75%

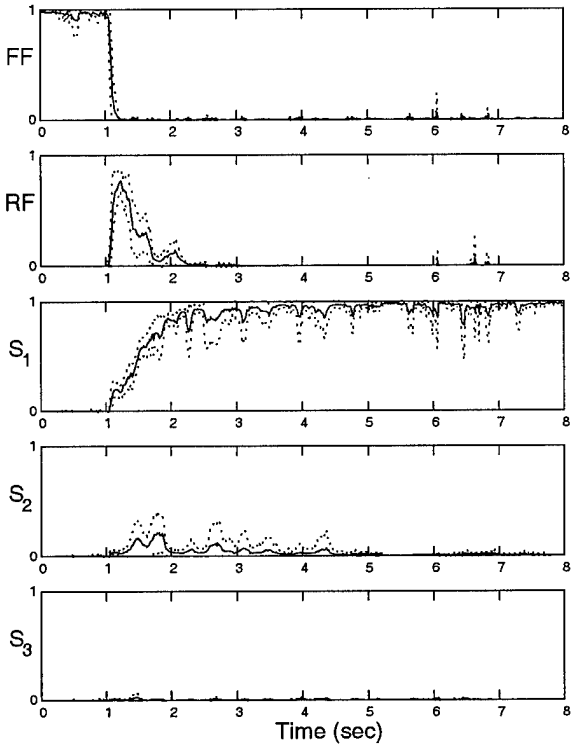


(d) Discretization Set: 40%, 60%, 80%

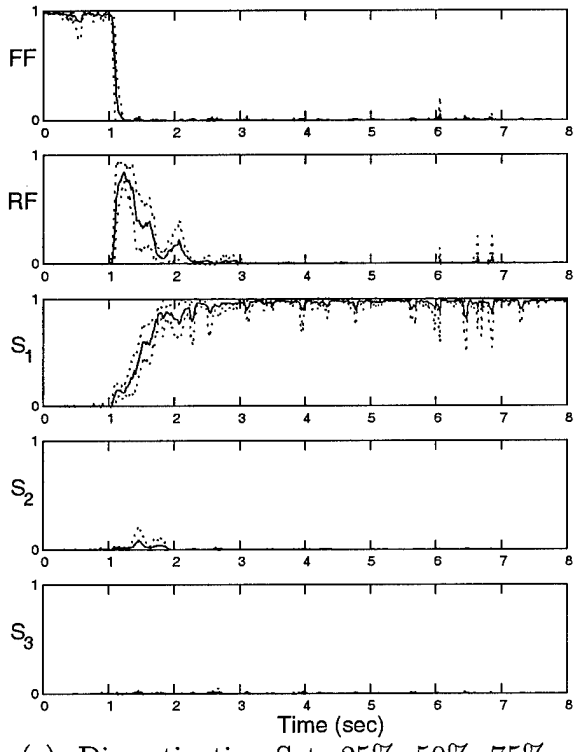
Figure 62. Probability Plot: Right Flaperon Failure, $\epsilon = 0\%$



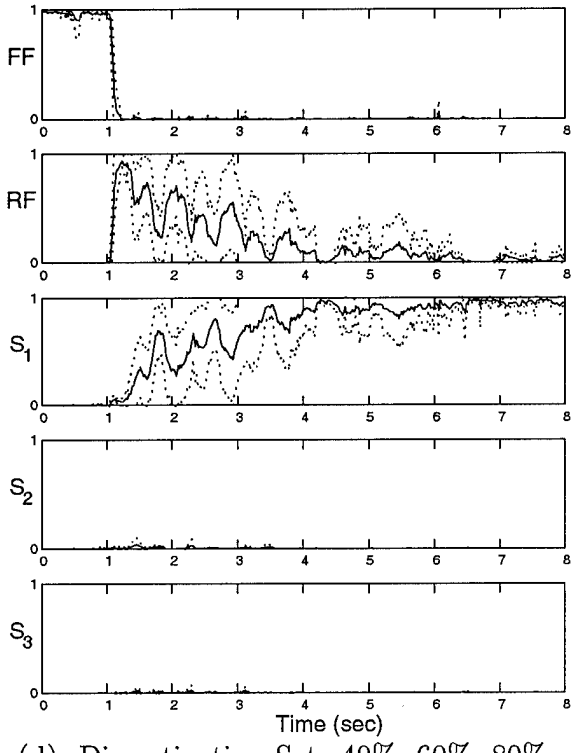
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

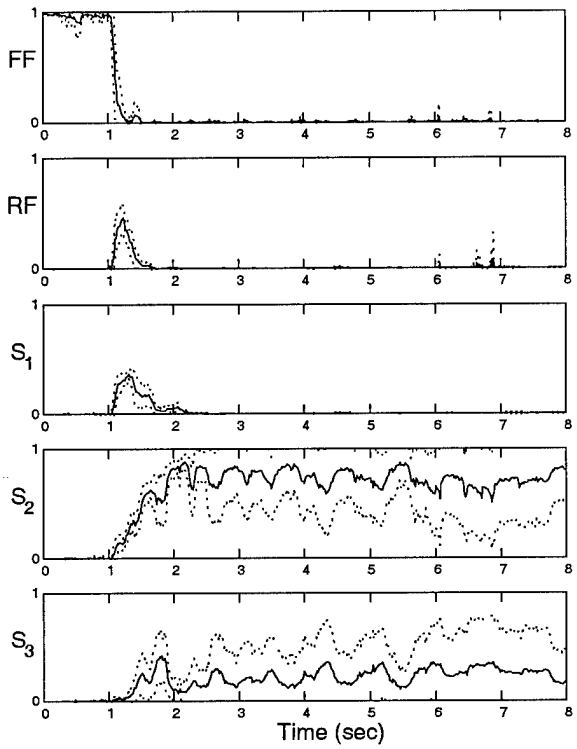


(c) Discretization Set: 25%, 50%, 75%

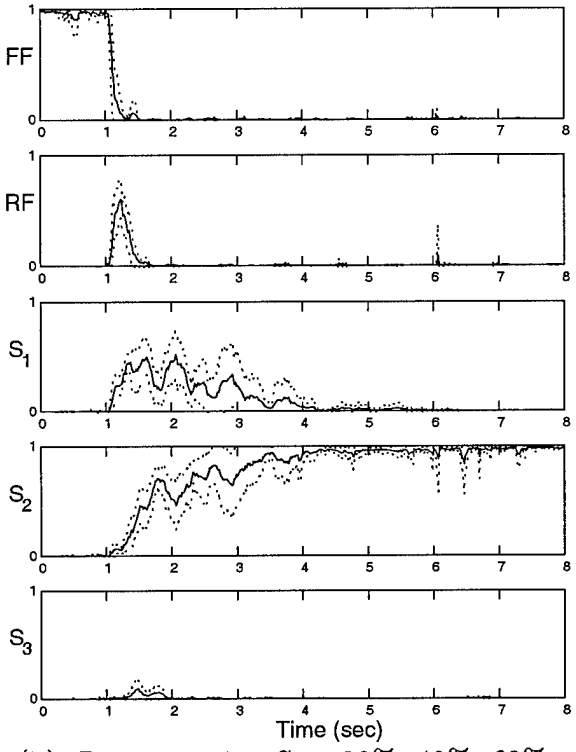


(d) Discretization Set: 40%, 60%, 80%

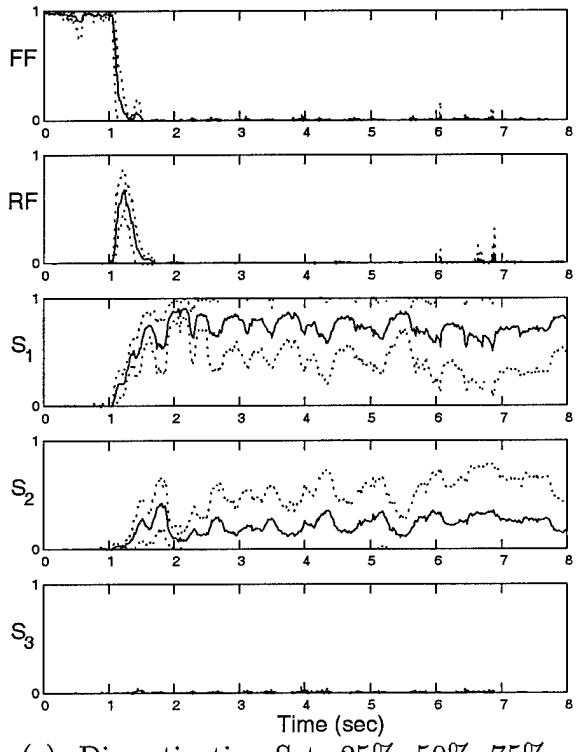
Figure 63. Probability Plot: Right Flaperon Failure, $\epsilon = 10\%$



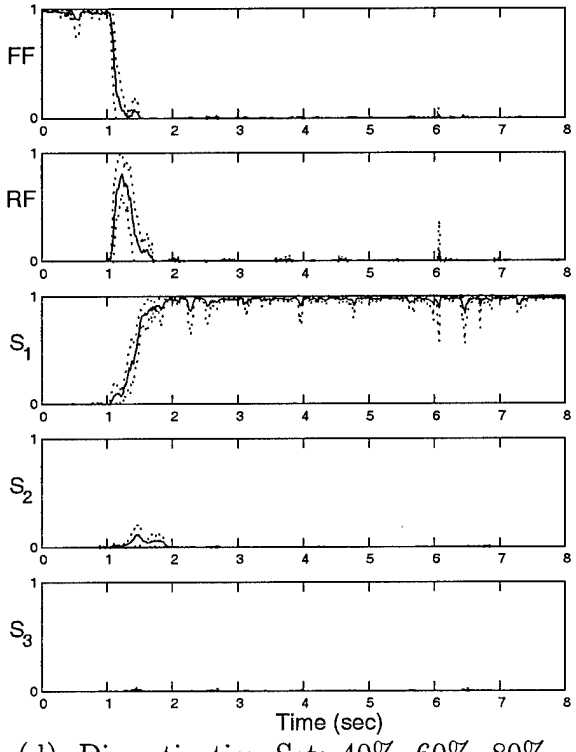
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

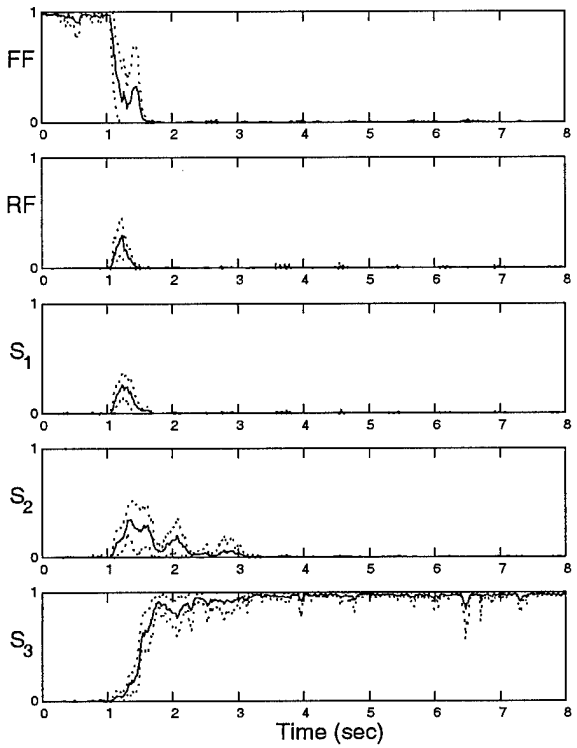


(c) Discretization Set: 25%, 50%, 75%

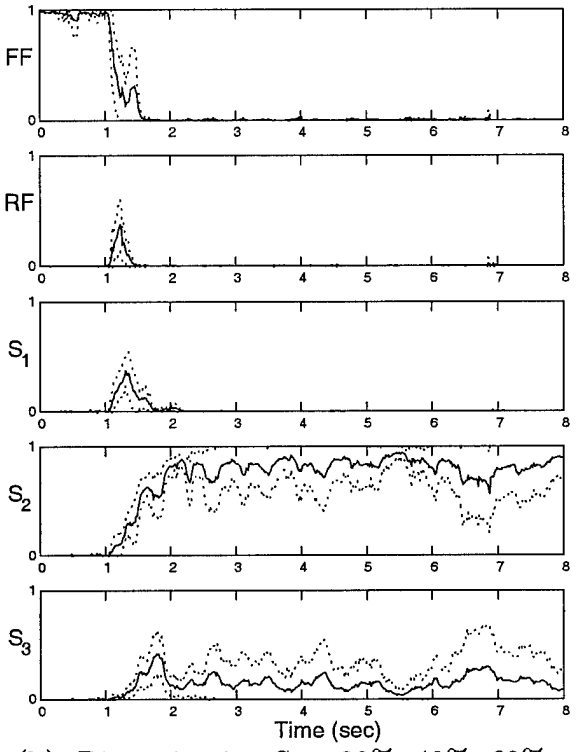


(d) Discretization Set: 40%, 60%, 80%

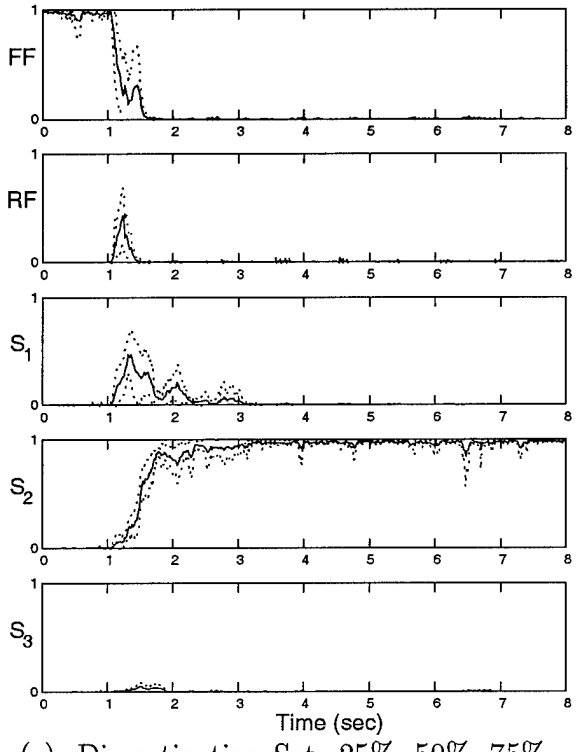
Figure 64. Probability Plot: Right Flaperon Failure, $\epsilon = 20\%$



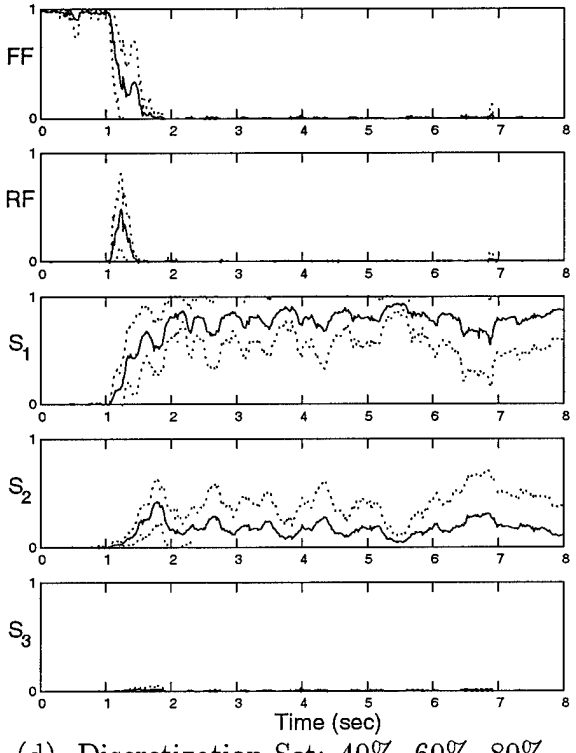
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

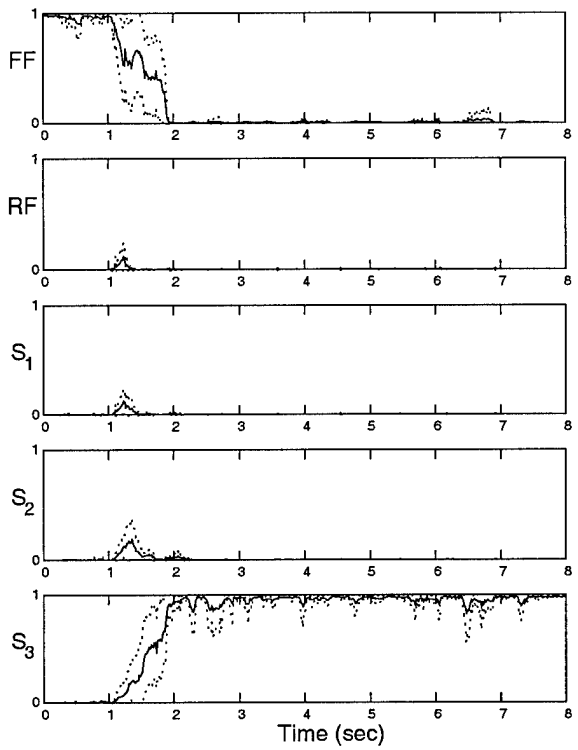


(c) Discretization Set: 25%, 50%, 75%

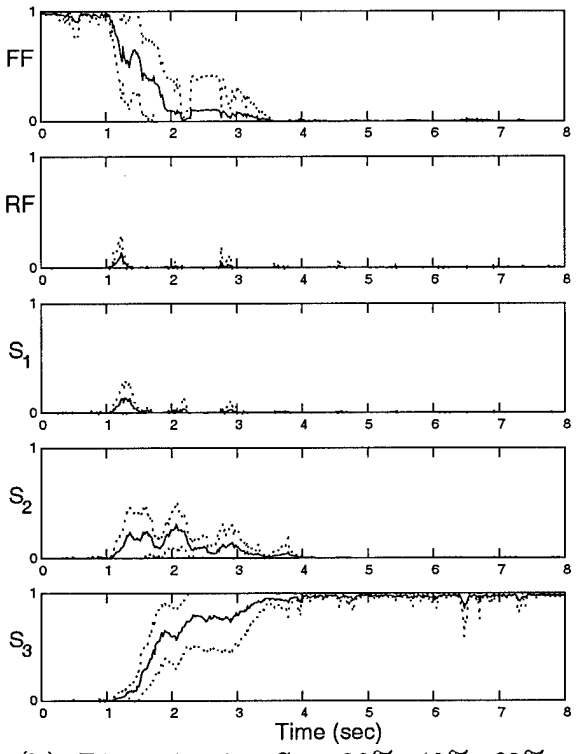


(d) Discretization Set: 40%, 60%, 80%

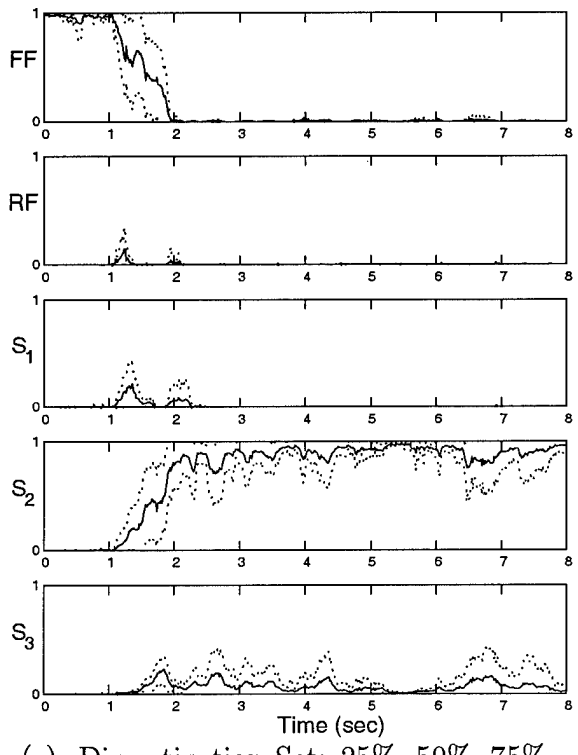
Figure 65. Probability Plot: Right Flaperon Failure, $\epsilon = 30\%$



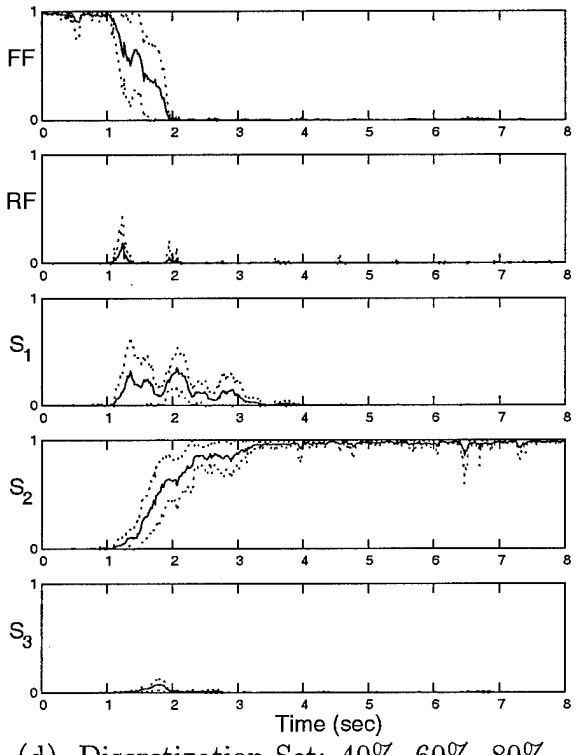
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

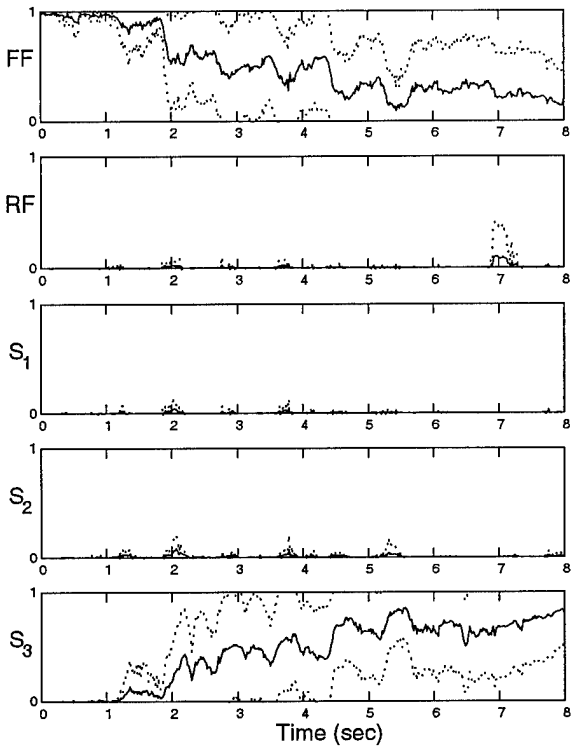


(c) Discretization Set: 25%, 50%, 75%

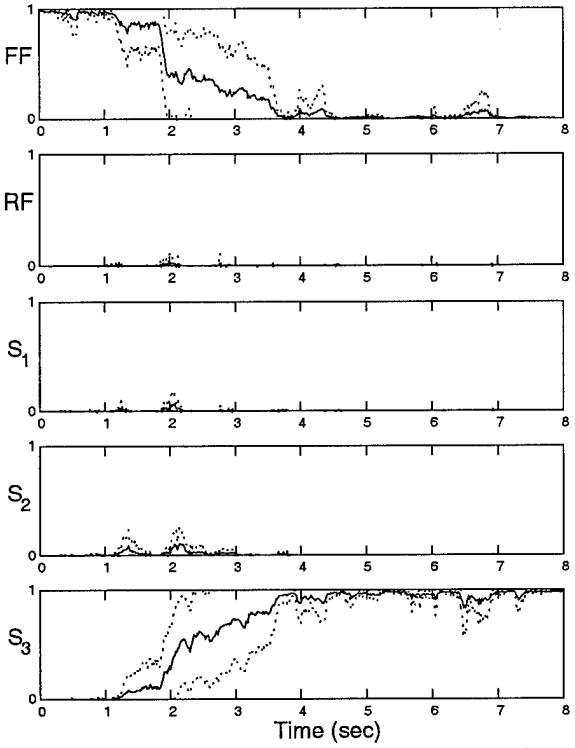


(d) Discretization Set: 40%, 60%, 80%

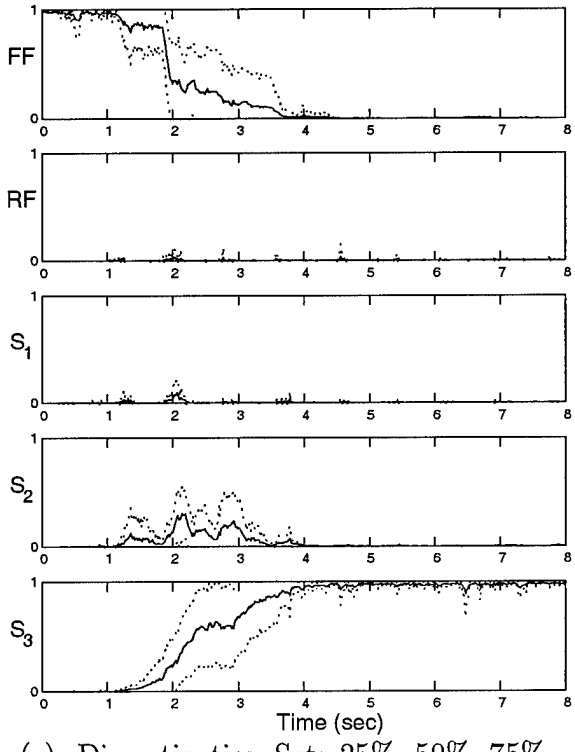
Figure 66. Probability Plot: Right Flaperon Failure, $\epsilon = 40\%$



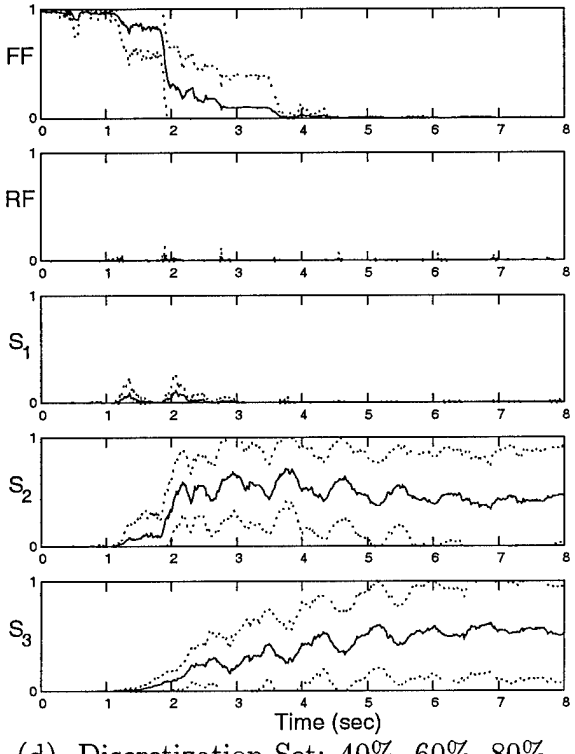
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

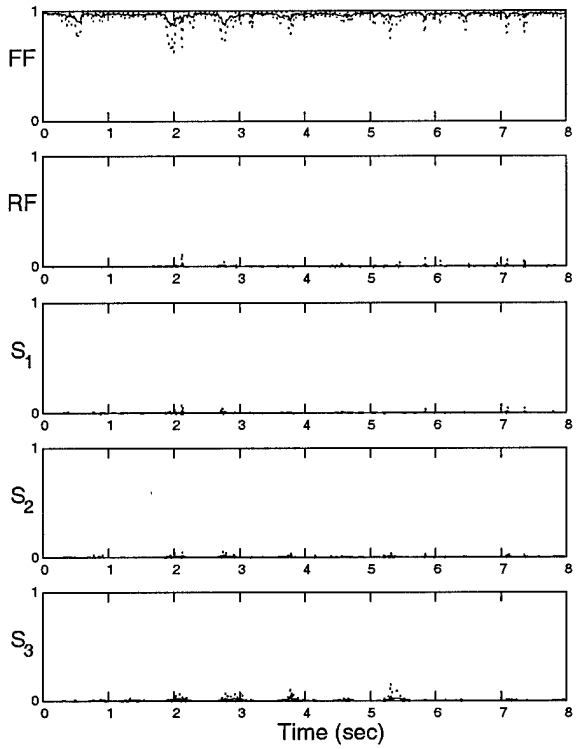


(c) Discretization Set: 25%, 50%, 75%

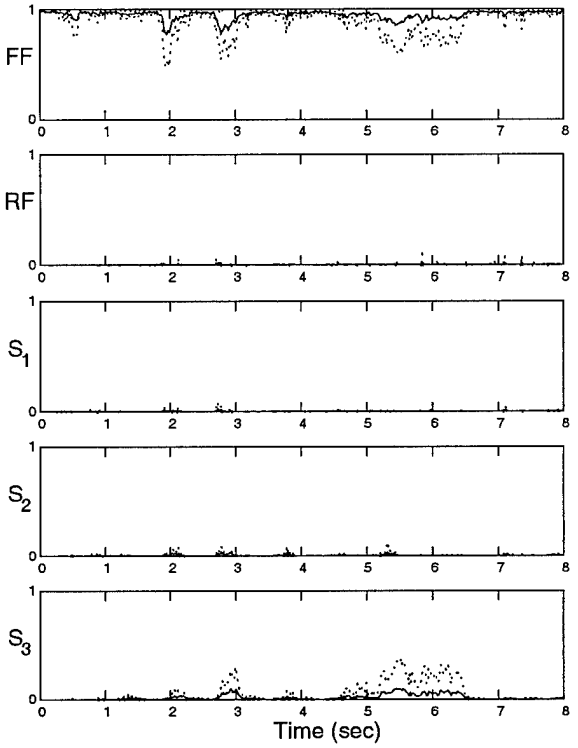


(d) Discretization Set: 40%, 60%, 80%

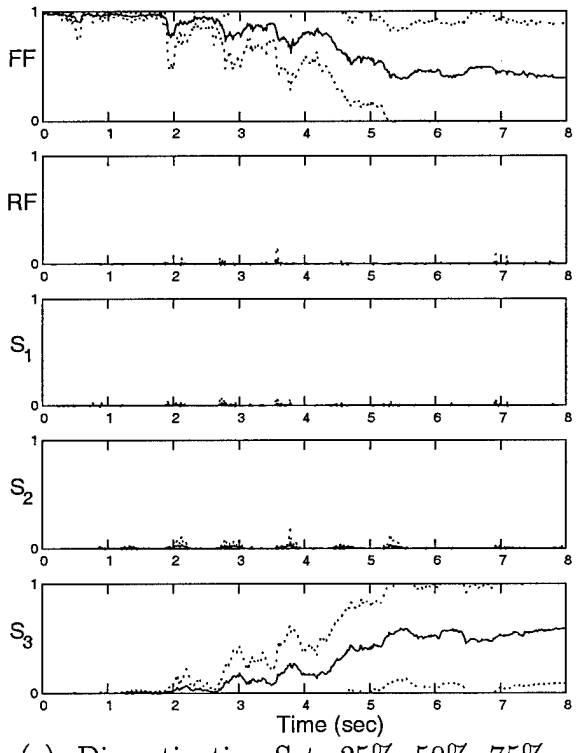
Figure 67. Probability Plot: Right Flaperon Failure, $\epsilon = 50\%$



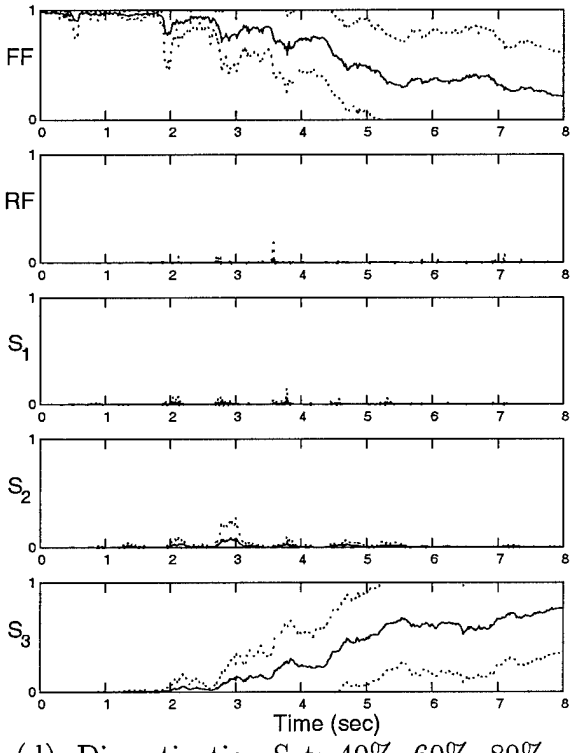
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

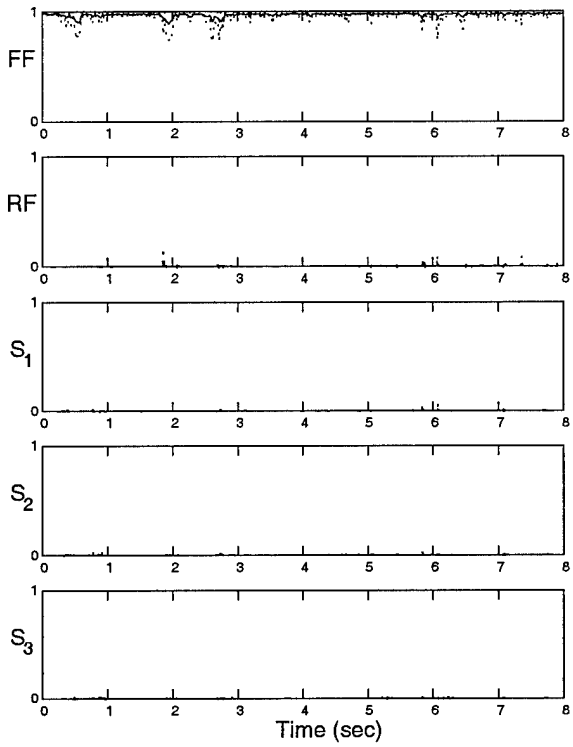


(c) Discretization Set: 25%, 50%, 75%

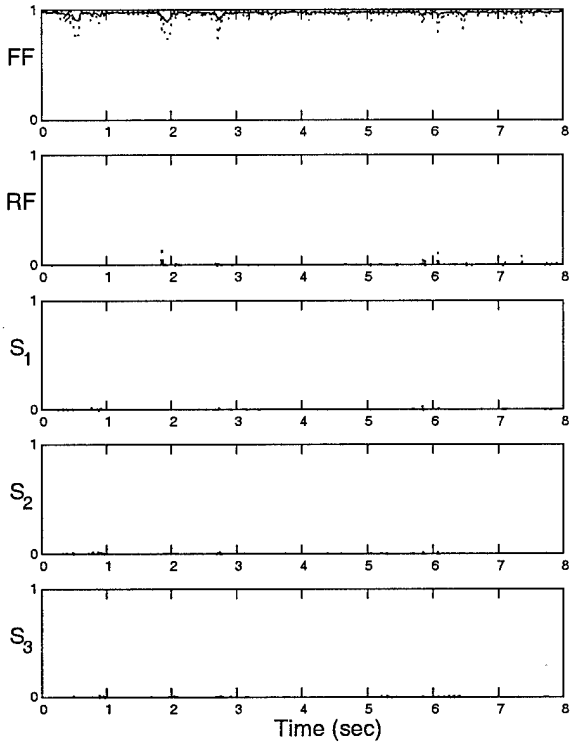


(d) Discretization Set: 40%, 60%, 80%

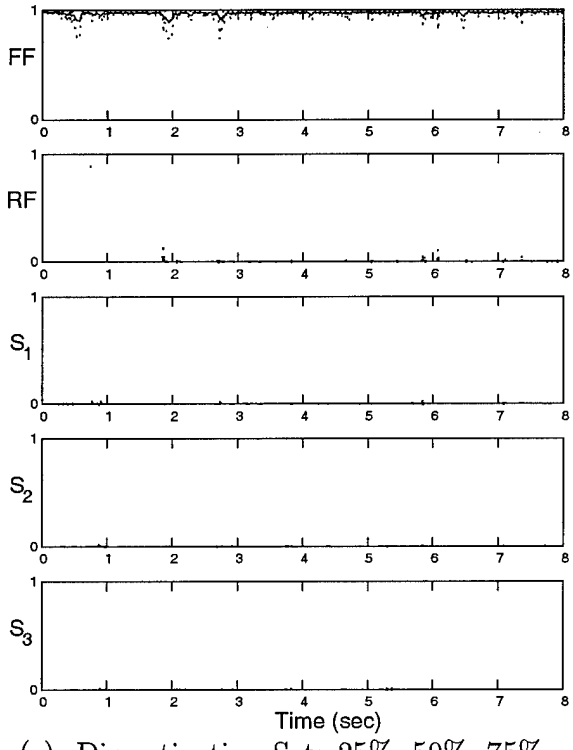
Figure 68. Probability Plot: Right Flaperon Failure, $\epsilon = 60\%$



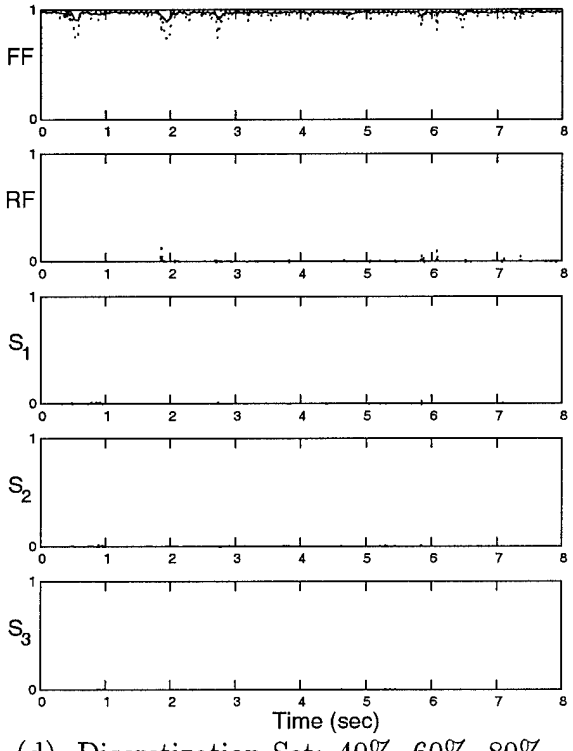
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

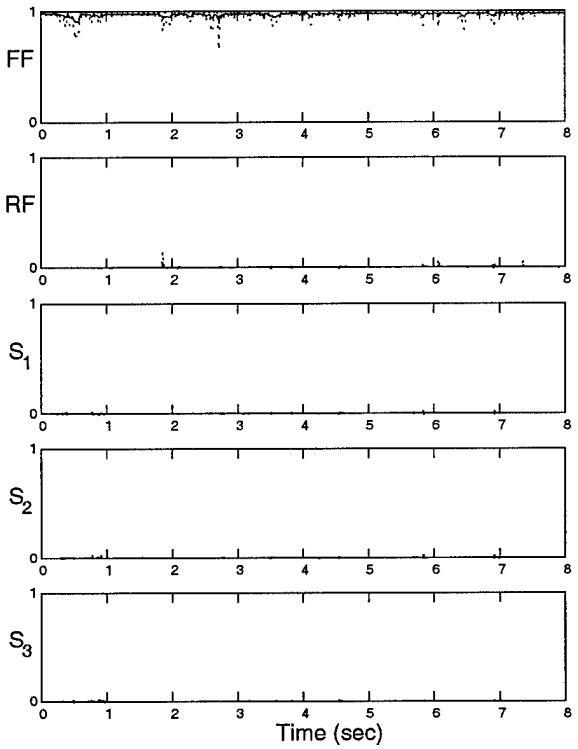


(c) Discretization Set: 25%, 50%, 75%

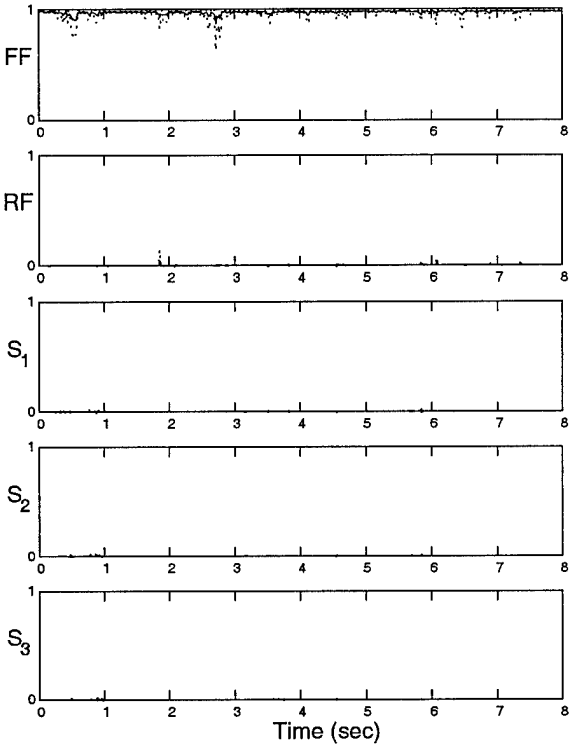


(d) Discretization Set: 40%, 60%, 80%

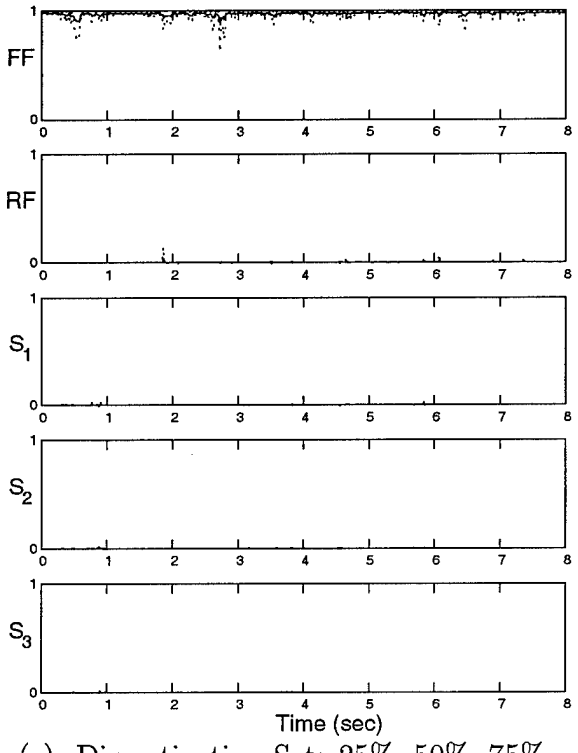
Figure 69. Probability Plot: Right Flaperon Failure, $\epsilon = 70\%$



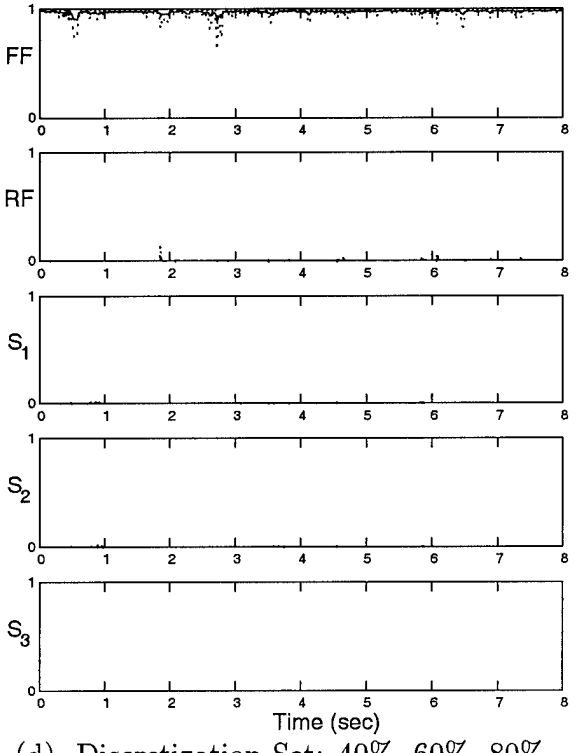
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

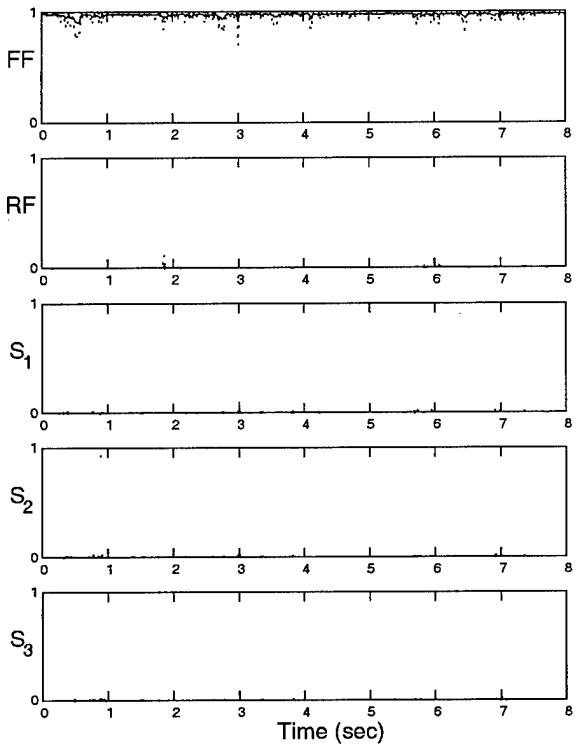


(c) Discretization Set: 25%, 50%, 75%

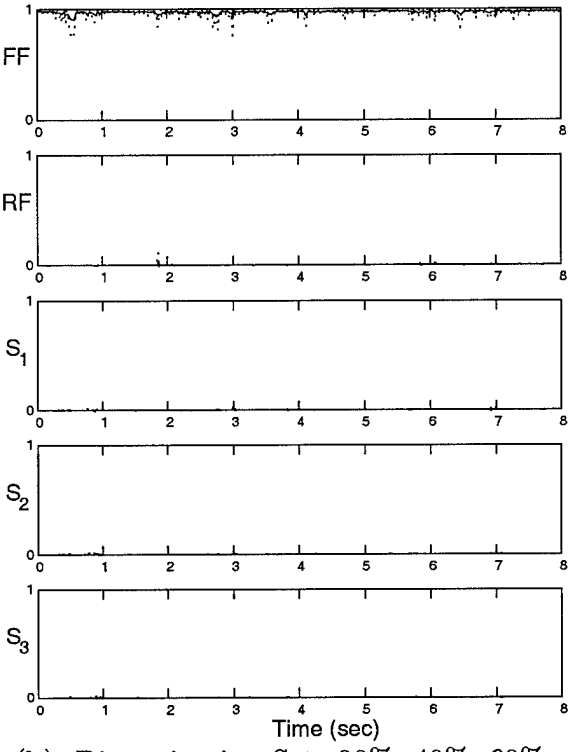


(d) Discretization Set: 40%, 60%, 80%

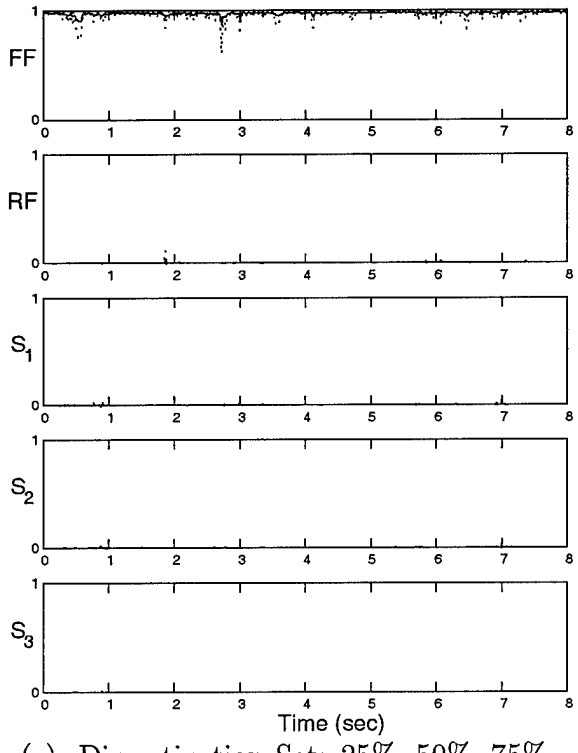
Figure 70. Probability Plot: Right Flaperon Failure, $\epsilon = 80\%$



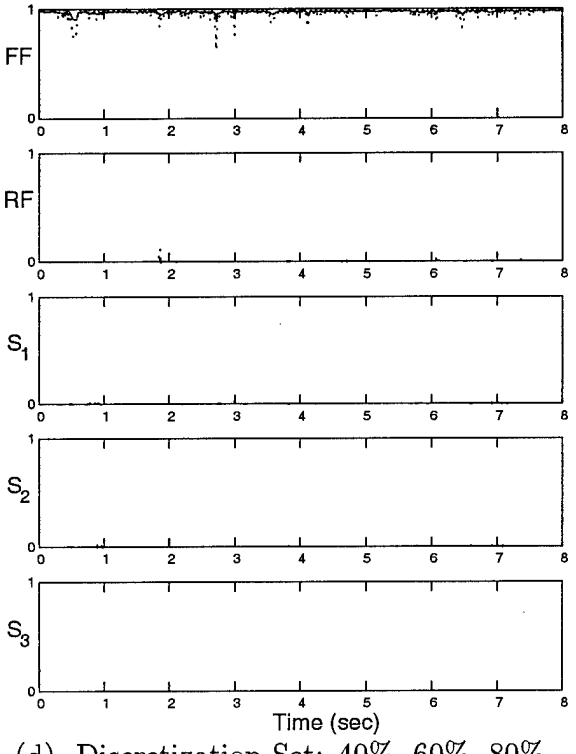
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

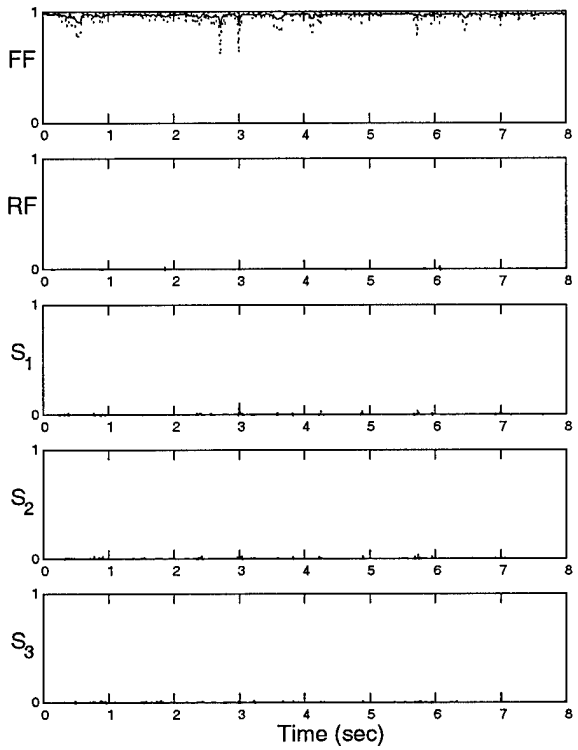


(c) Discretization Set: 25%, 50%, 75%

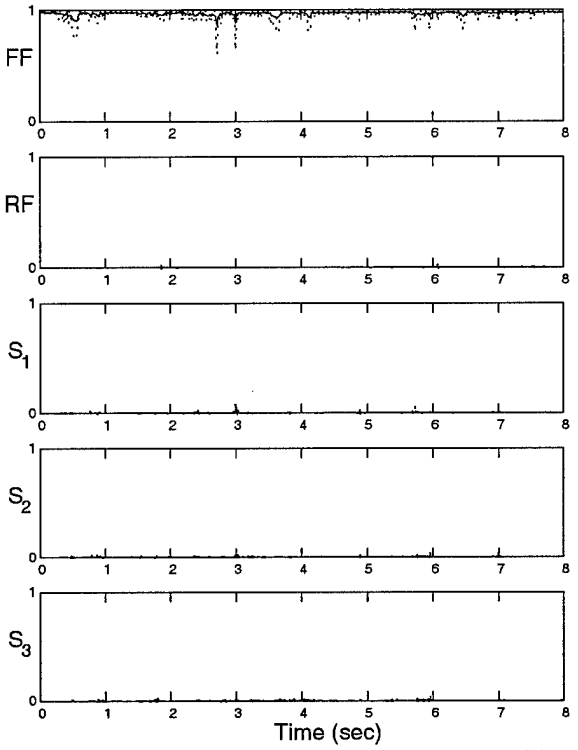


(d) Discretization Set: 40%, 60%, 80%

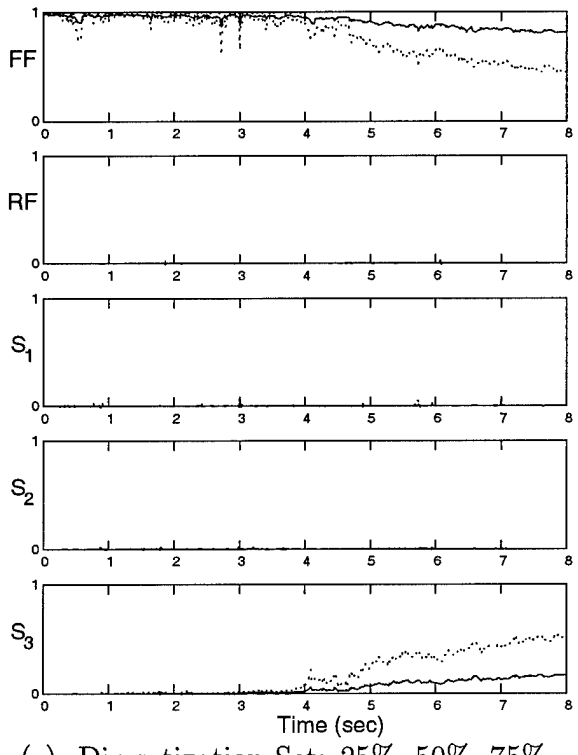
Figure 71. Probability Plot: Right Flaperon Failure, $\epsilon = 90\%$



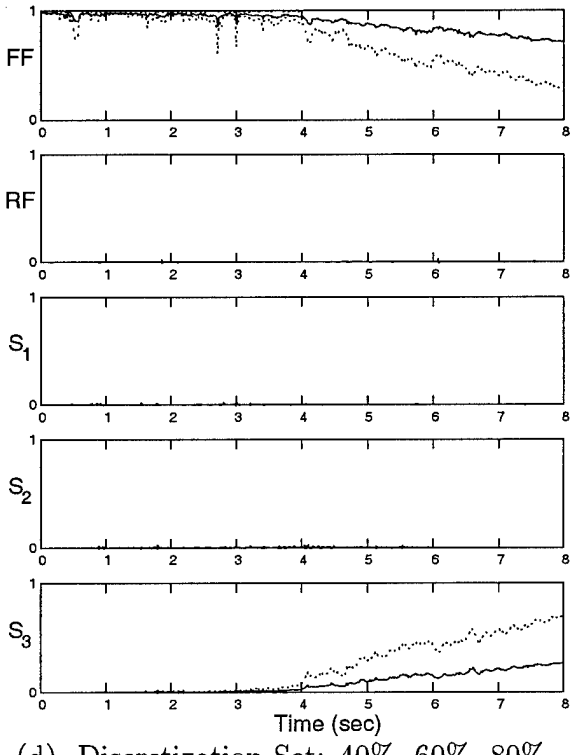
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



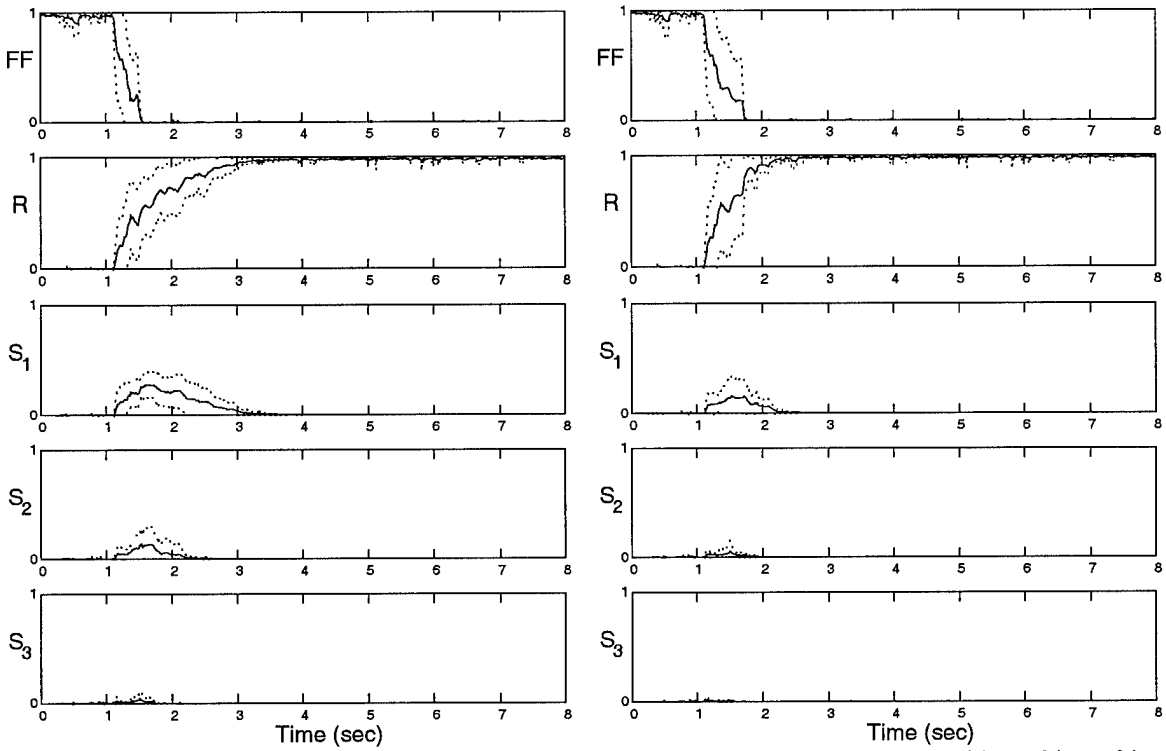
(d) Discretization Set: 40%, 60%, 80%

Figure 72. Probability Plot: Right Flaperon Failure, $\epsilon = 100\%$

B.4 Rudder

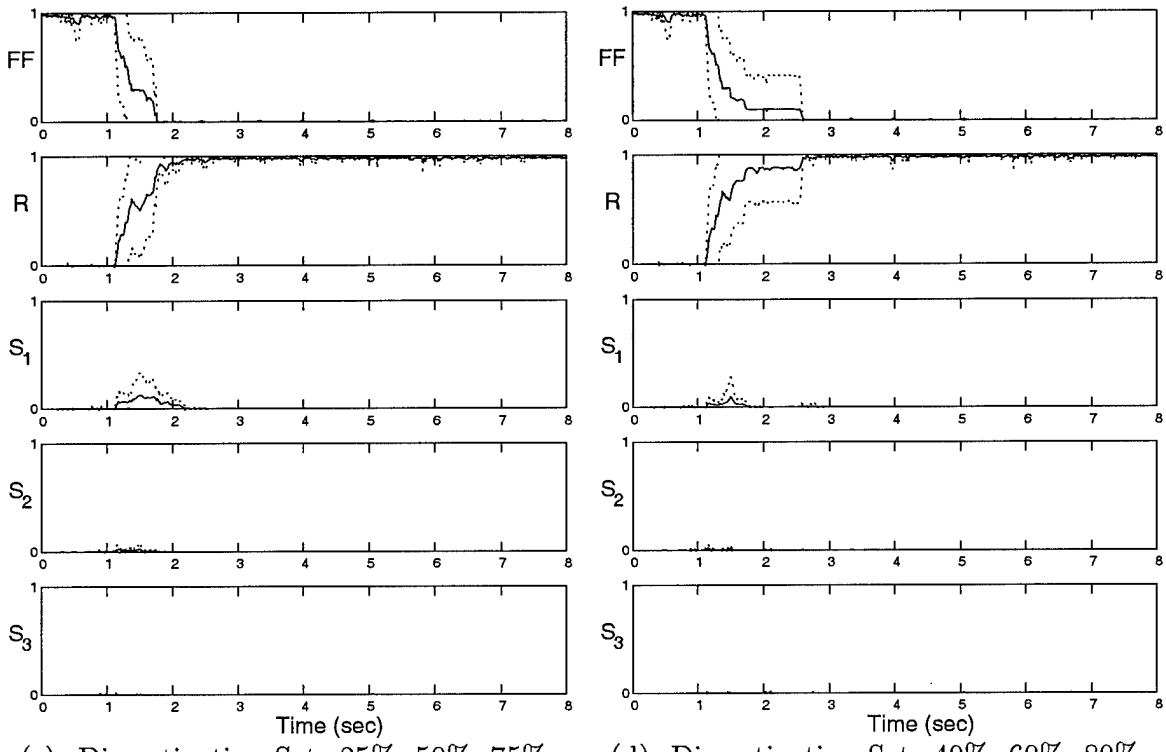
The corresponding figures for each effectiveness value of the rudder failure are given as

Figure	**Rudder Failure Effectiveness** ϵ_{true}	**Page**
73	0%	172
74	10%	173
75	20%	174
76	30%	175
77	40%	176
78	50%	177
79	60%	178
80	70%	179
81	80%	180
82	90%	181
83	100%	182



(a) Discretization Set: 10%, 25%, 50%

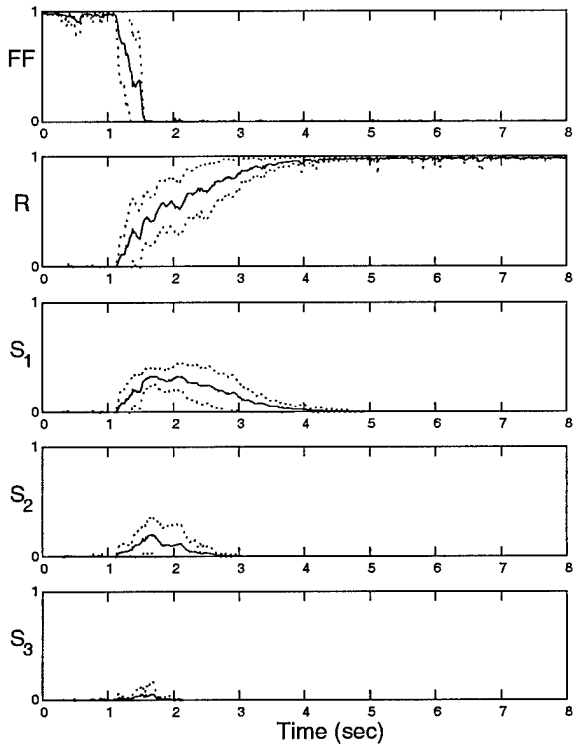
(b) Discretization Set: 20%, 40%, 60%



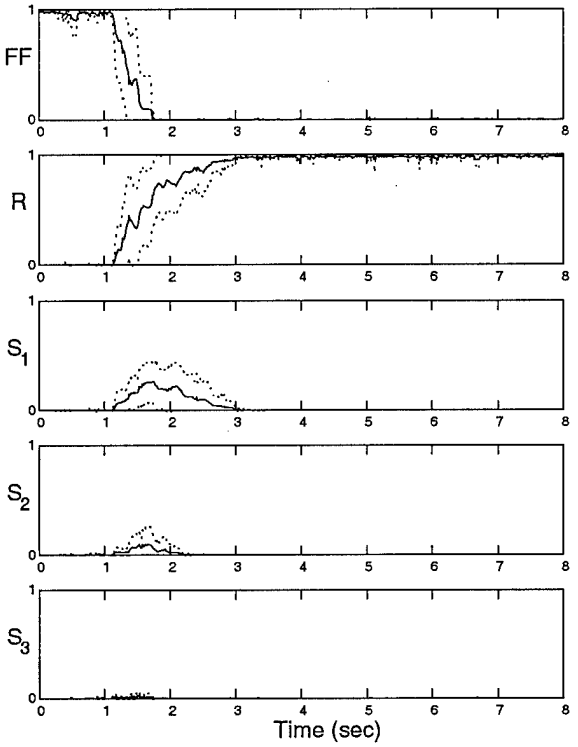
(c) Discretization Set: 25%, 50%, 75%

(d) Discretization Set: 40%, 60%, 80%

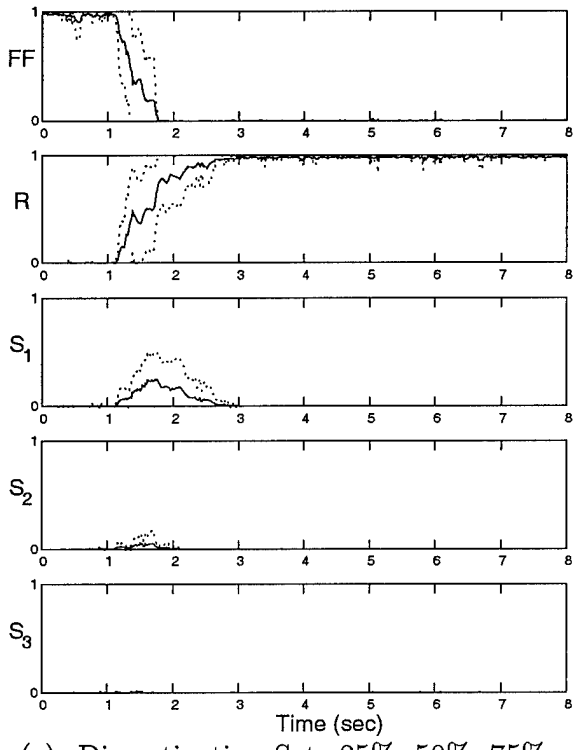
Figure 73. Probability Plot: Rudder Failure, $\epsilon = 0\%$



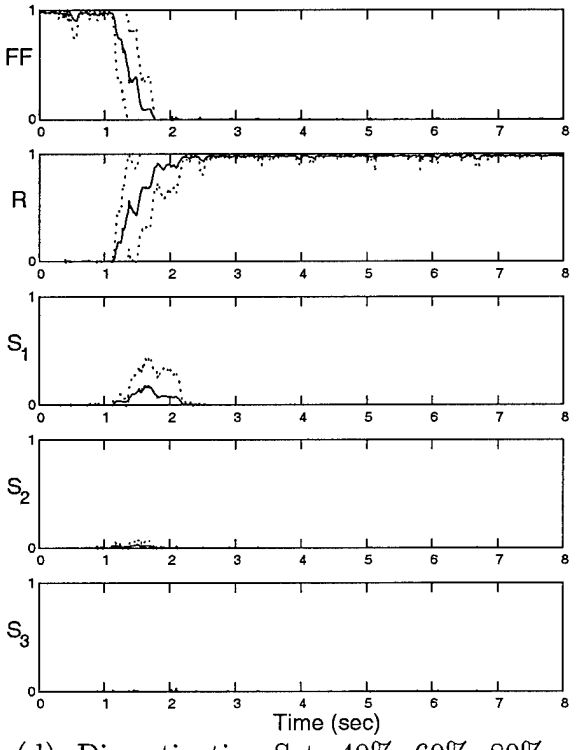
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

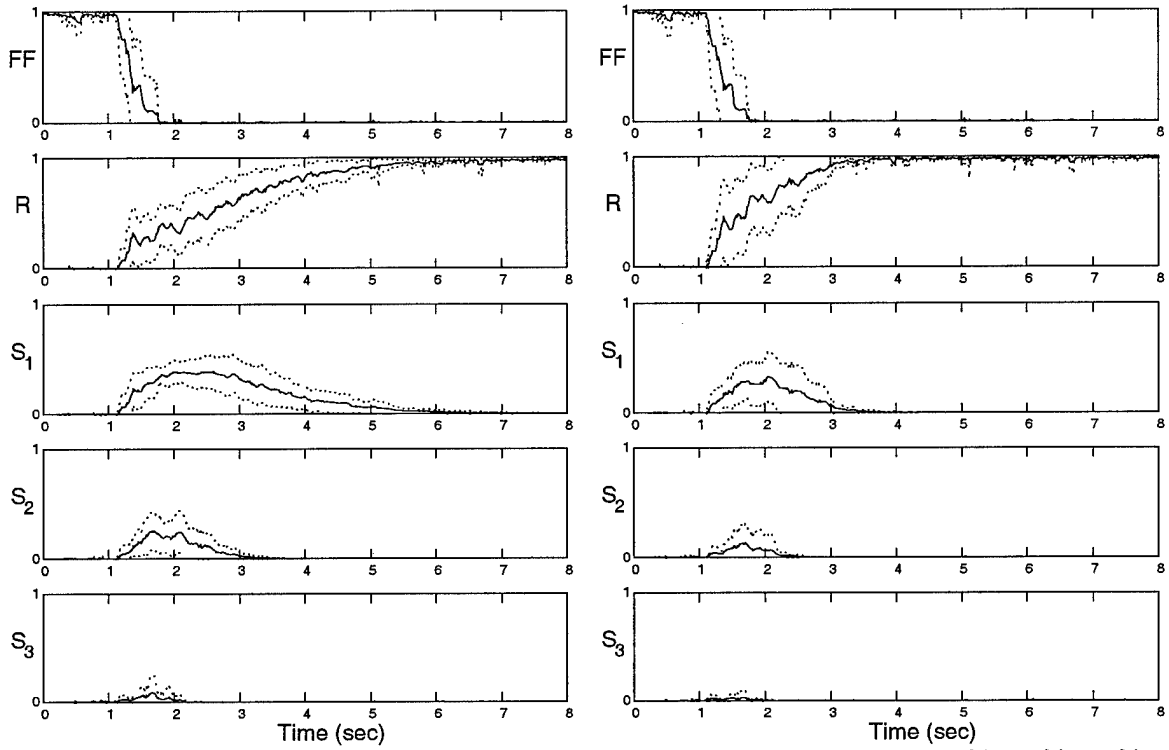


(c) Discretization Set: 25%, 50%, 75%



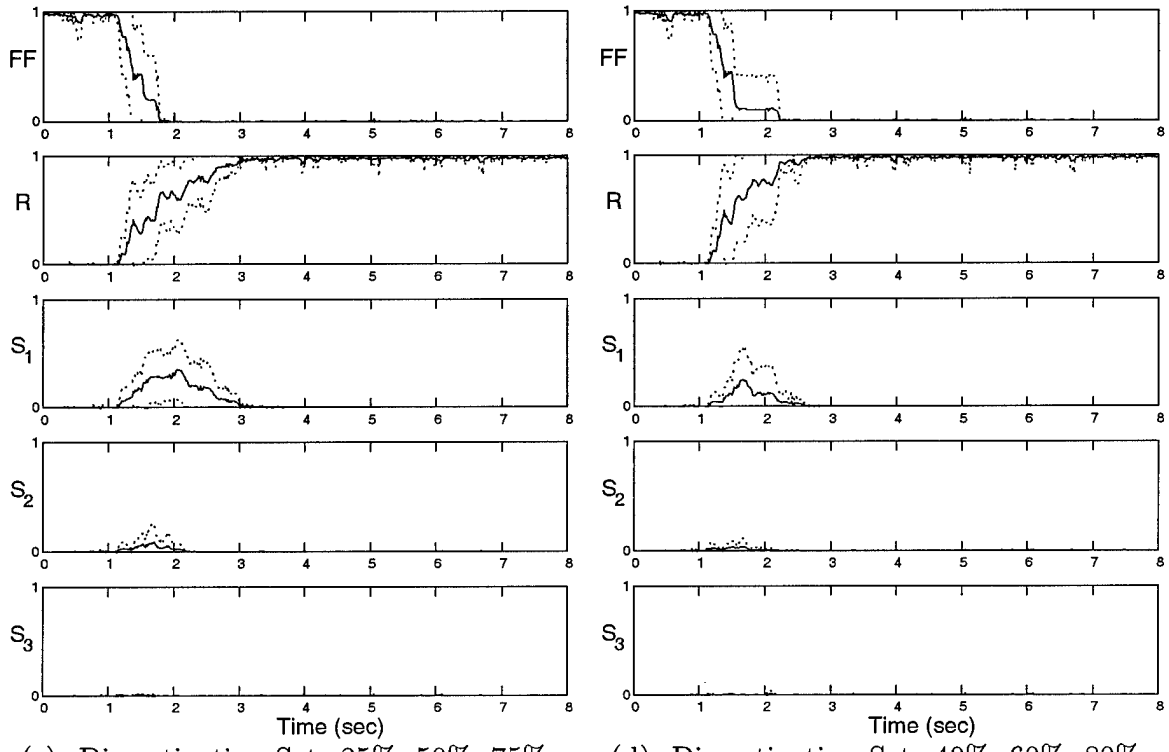
(d) Discretization Set: 40%, 60%, 80%

Figure 74. Probability Plot: Rudder Failure, $\epsilon = 10\%$



(a) Discretization Set: 10%, 25%, 50%

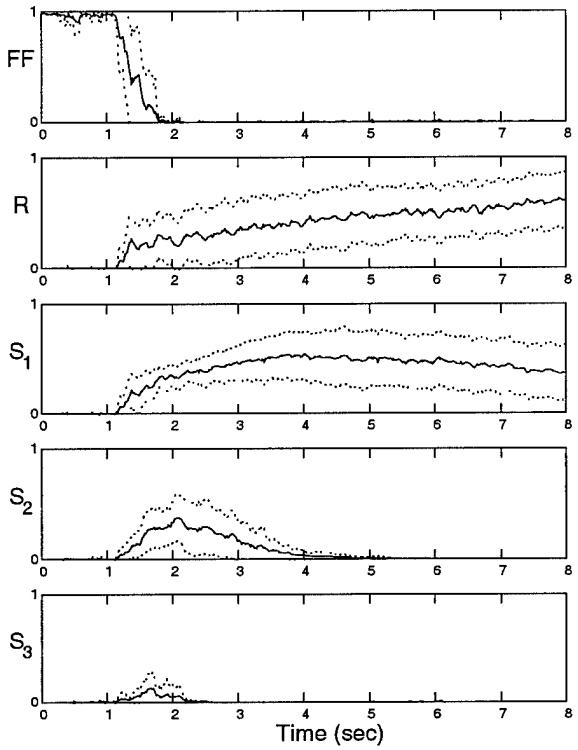
(b) Discretization Set: 20%, 40%, 60%



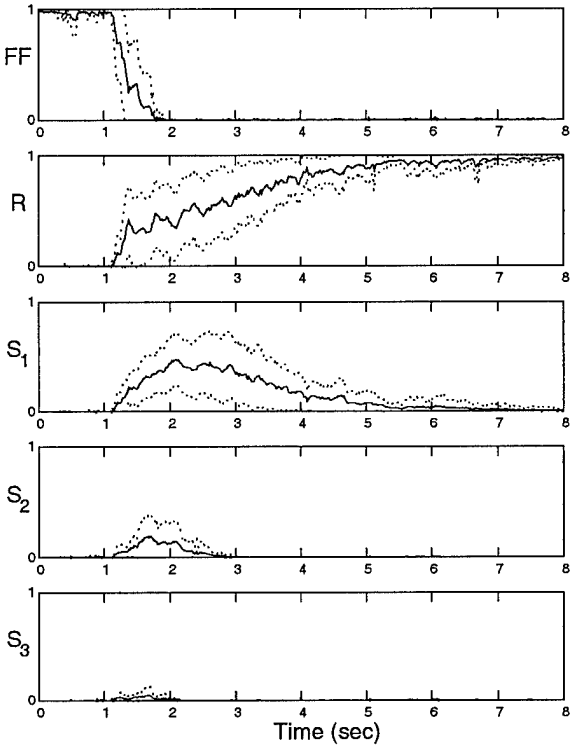
(c) Discretization Set: 25%, 50%, 75%

(d) Discretization Set: 40%, 60%, 80%

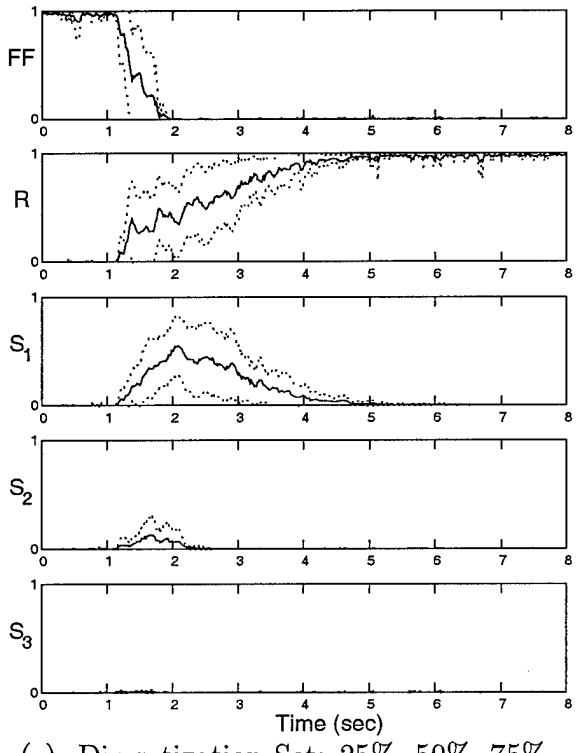
Figure 75. Probability Plot: Rudder Failure, $\epsilon = 20\%$



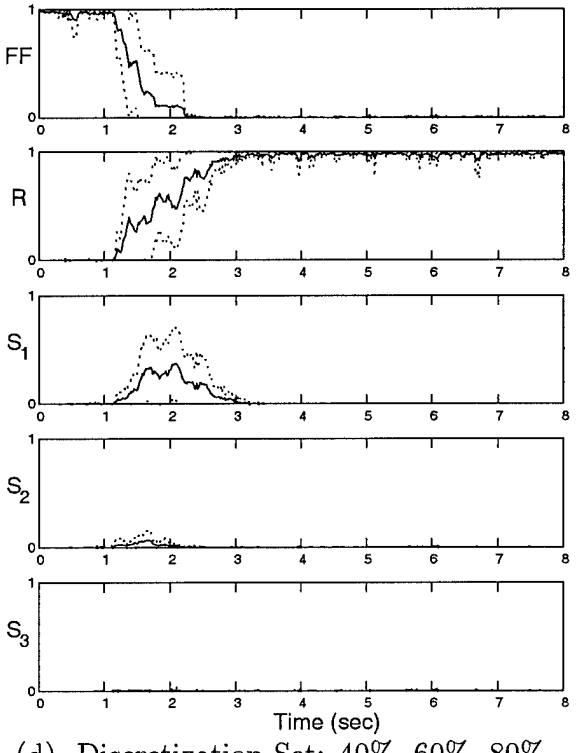
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

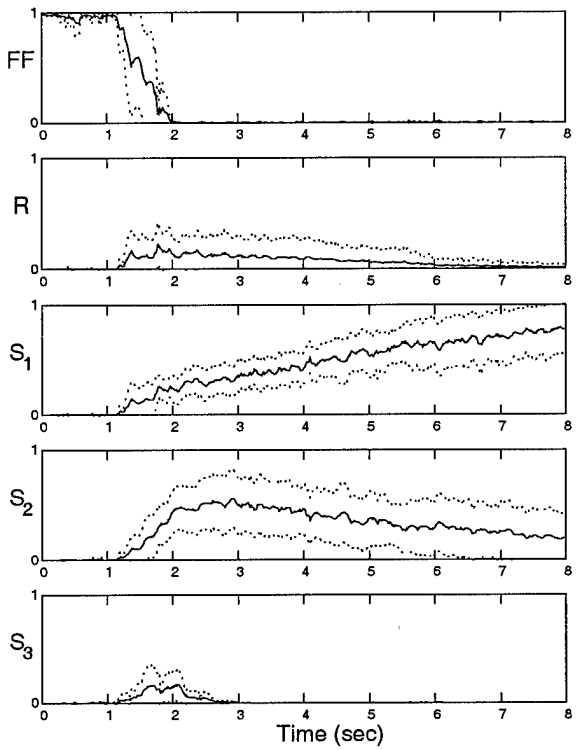


(c) Discretization Set: 25%, 50%, 75%

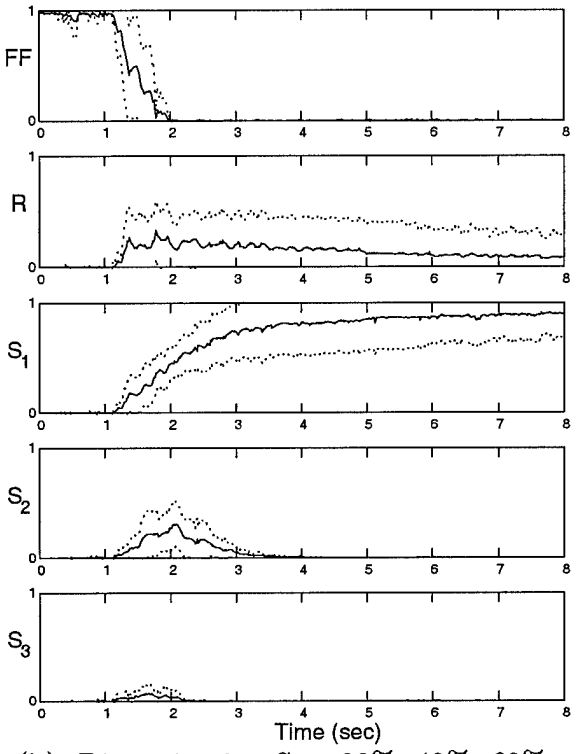


(d) Discretization Set: 40%, 60%, 80%

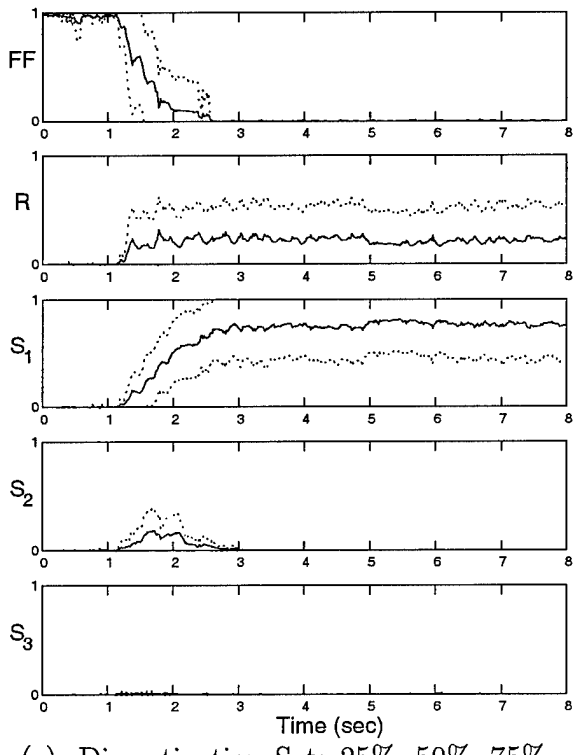
Figure 76. Probability Plot: Rudder Failure, $\epsilon = 30\%$



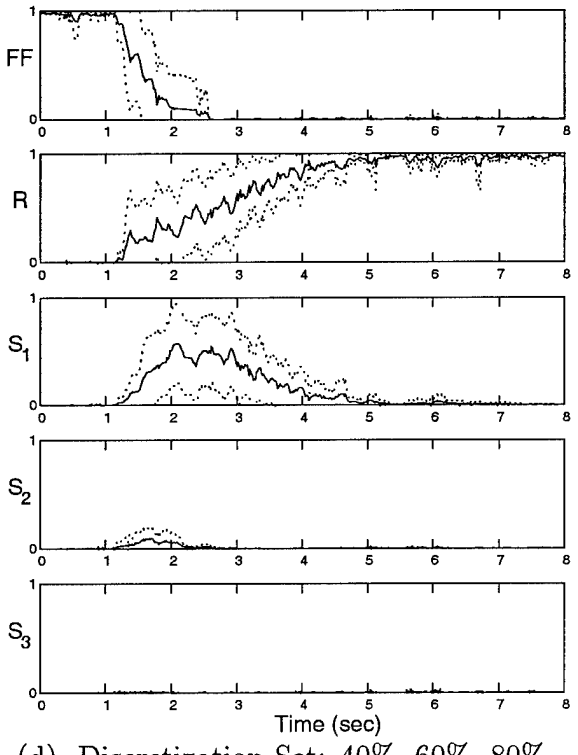
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

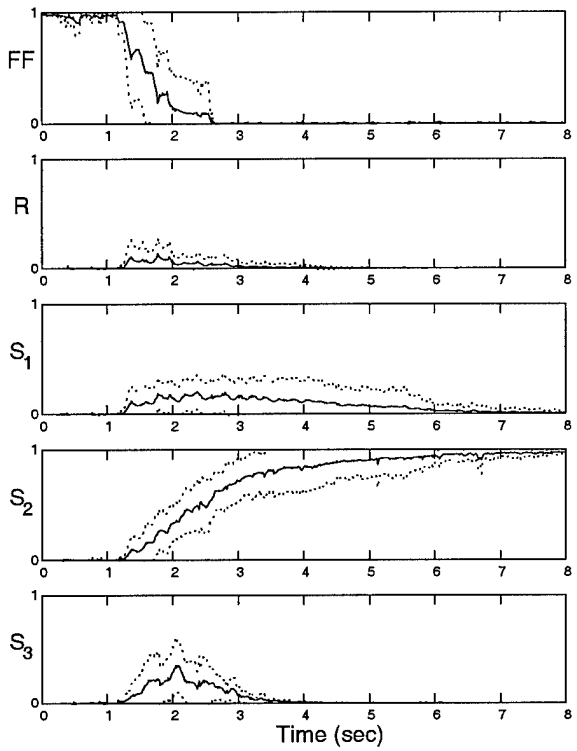


(c) Discretization Set: 25%, 50%, 75%

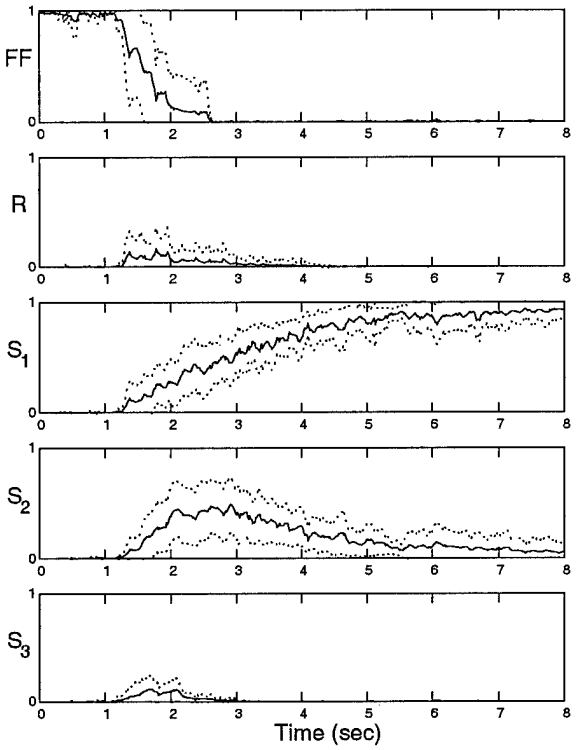


(d) Discretization Set: 40%, 60%, 80%

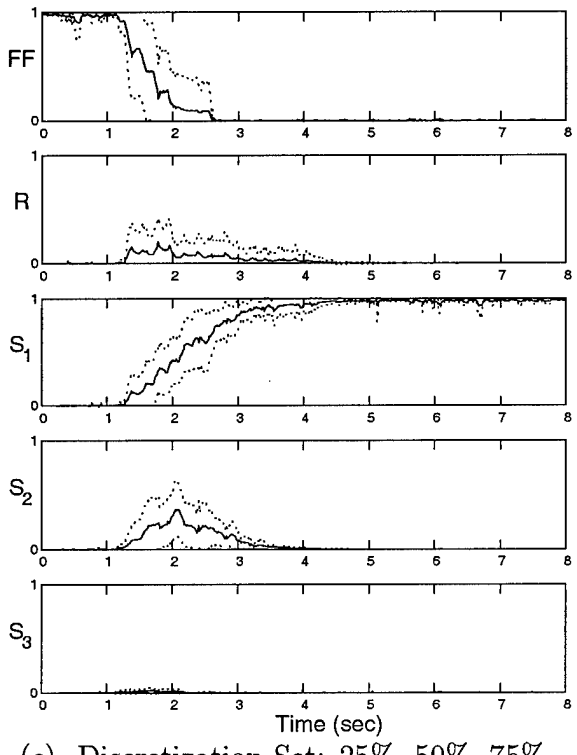
Figure 77. Probability Plot: Rudder Failure, $\epsilon = 40\%$



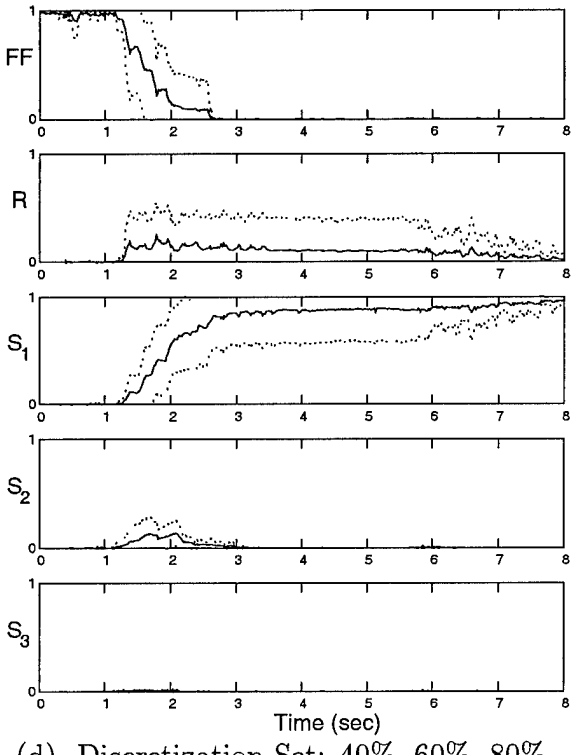
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

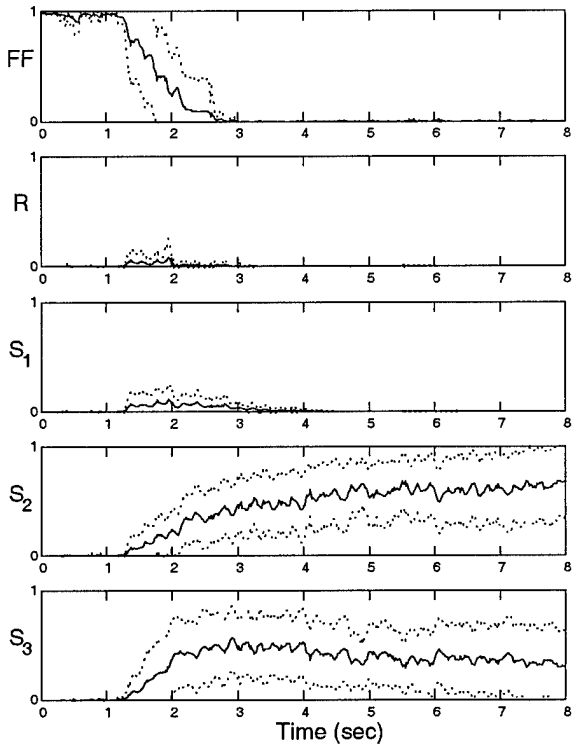


(c) Discretization Set: 25%, 50%, 75%

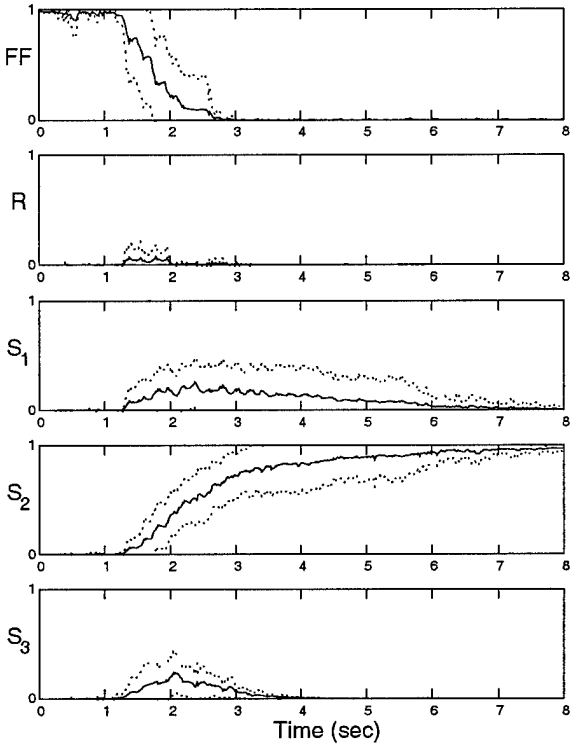


(d) Discretization Set: 40%, 60%, 80%

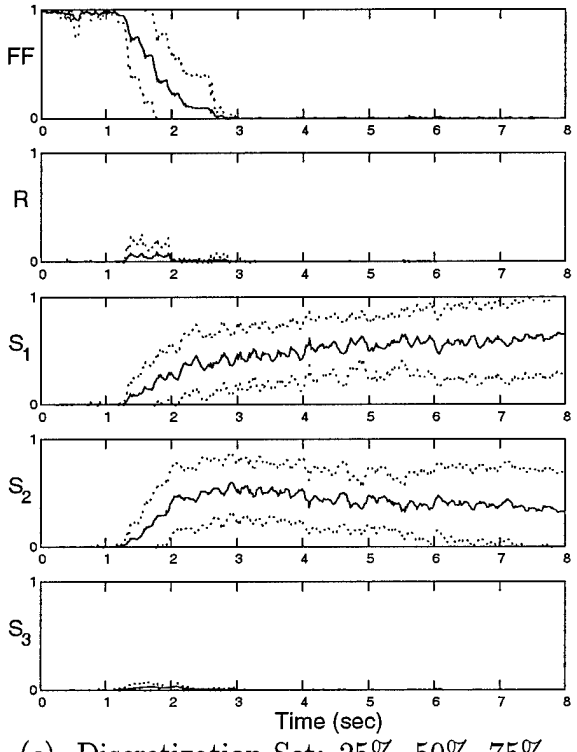
Figure 78. Probability Plot: Rudder Failure, $\epsilon = 50\%$



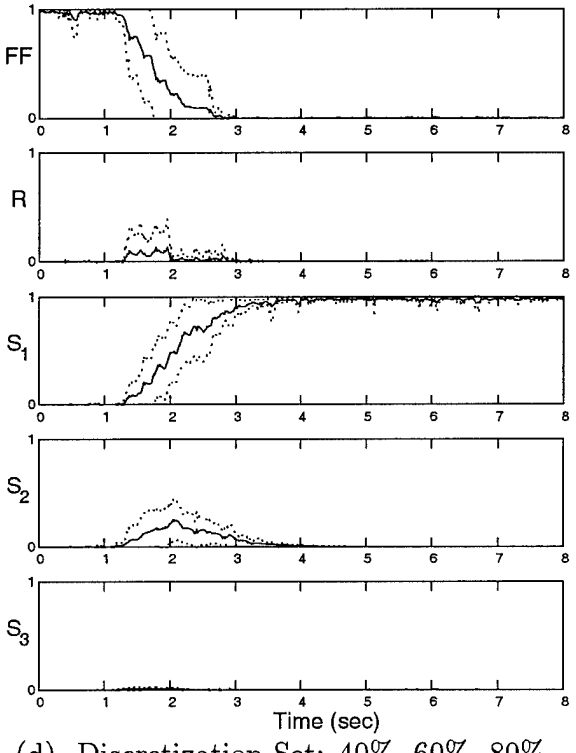
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

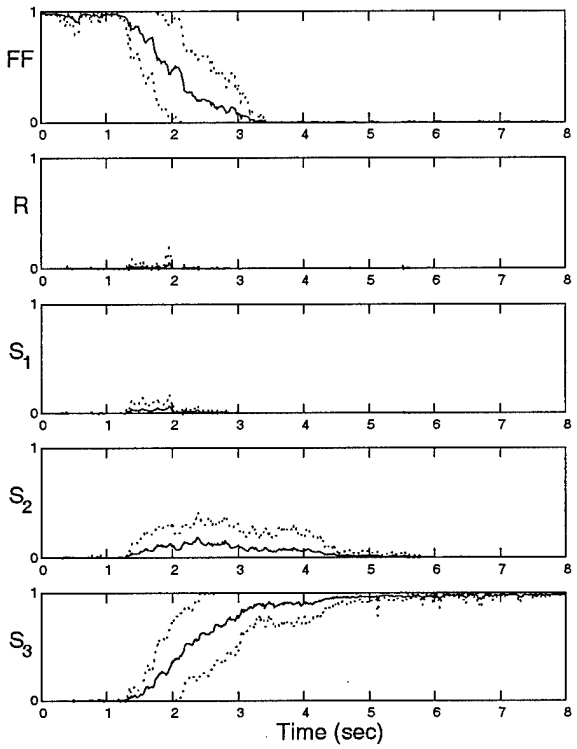


(c) Discretization Set: 25%, 50%, 75%

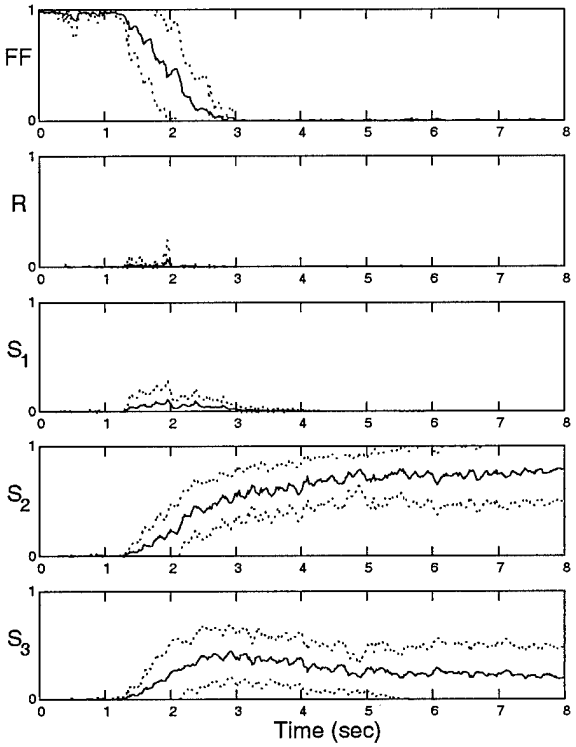


(d) Discretization Set: 40%, 60%, 80%

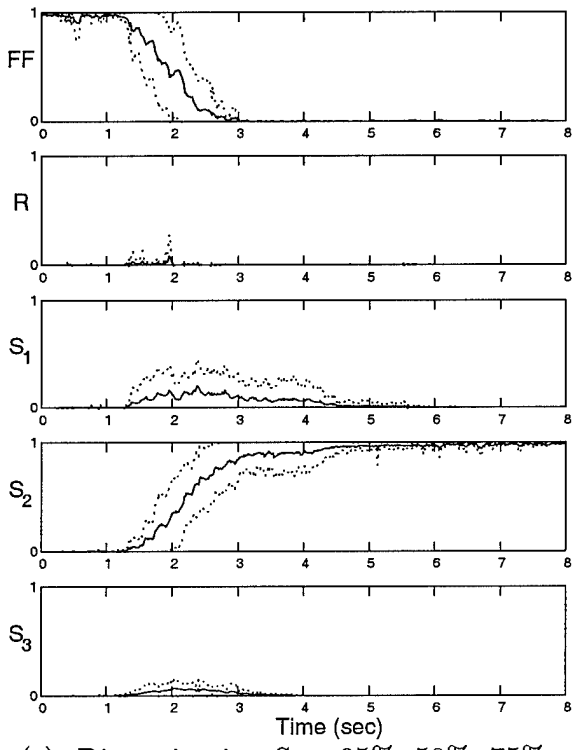
Figure 79. Probability Plot: Rudder Failure, $\epsilon = 60\%$



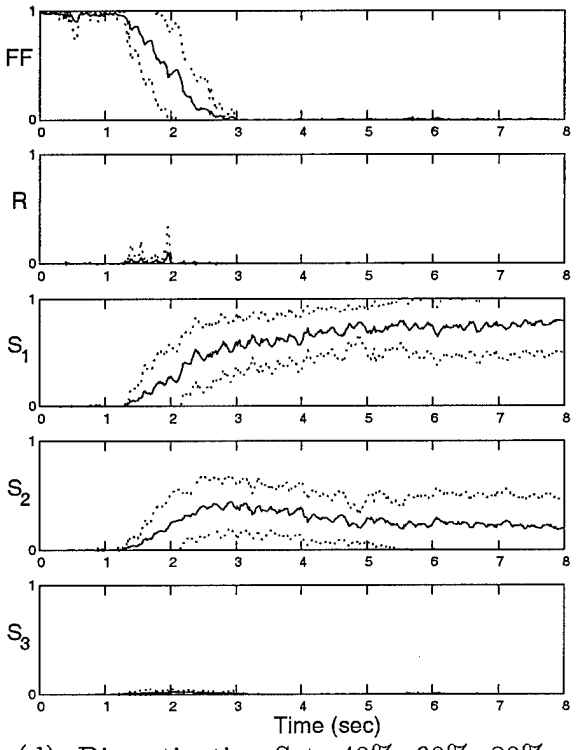
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

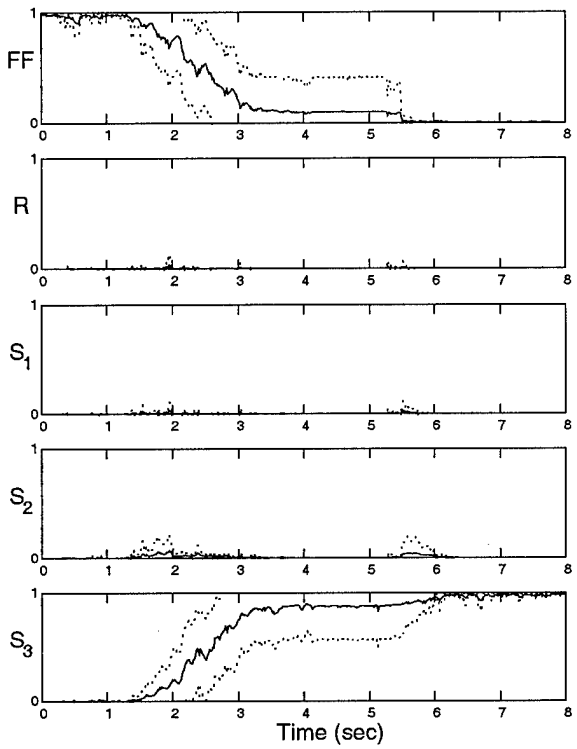


(c) Discretization Set: 25%, 50%, 75%

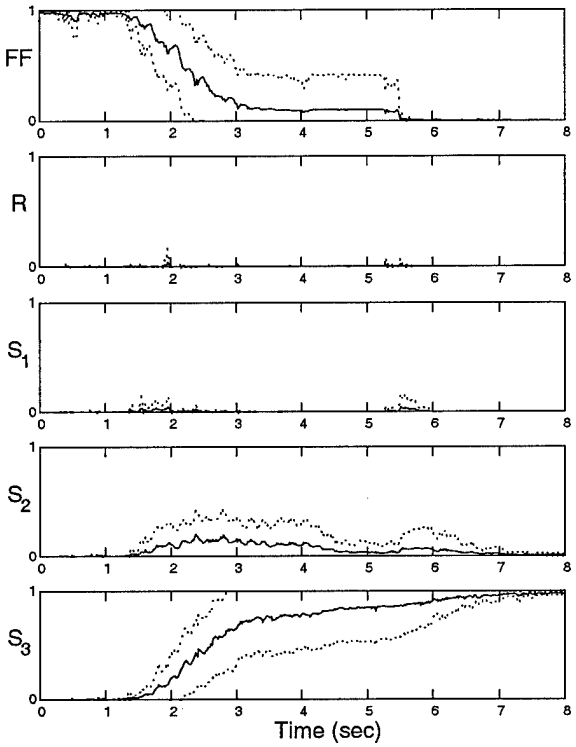


(d) Discretization Set: 40%, 60%, 80%

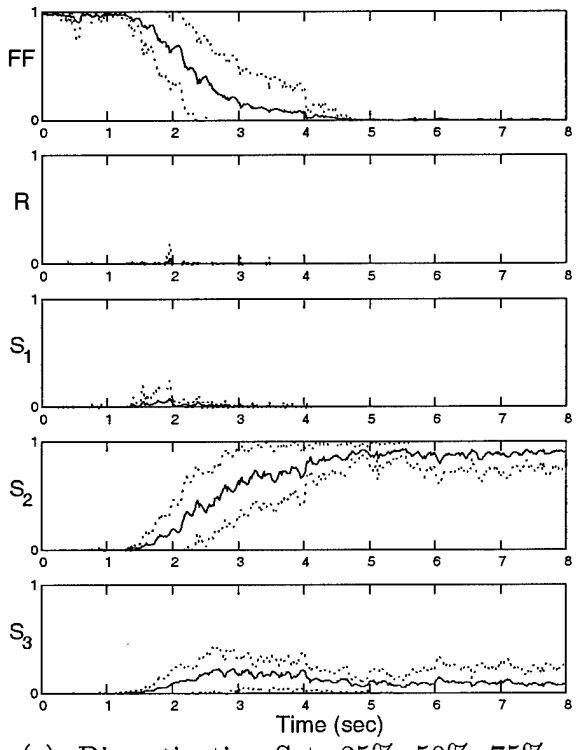
Figure 80. Probability Plot: Rudder Failure, $\epsilon = 70\%$



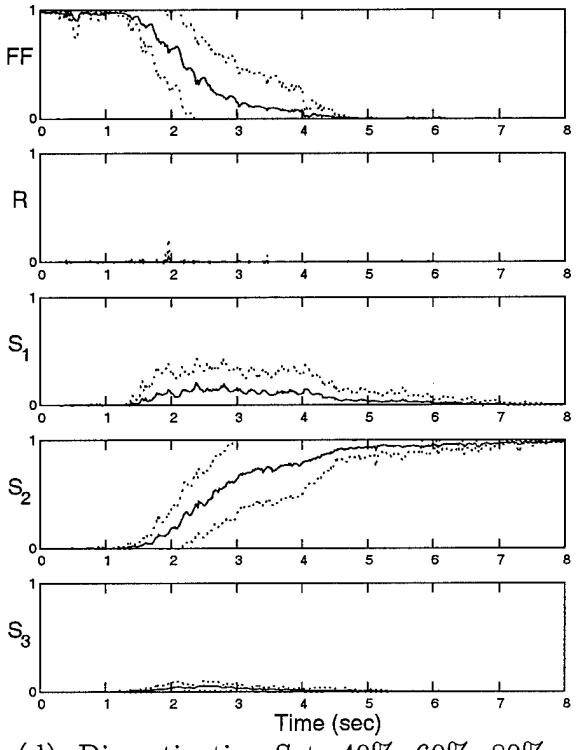
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

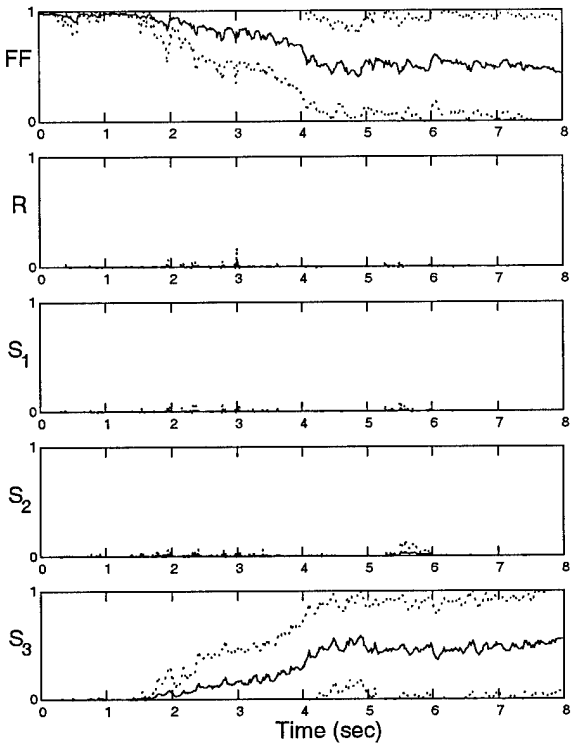


(c) Discretization Set: 25%, 50%, 75%

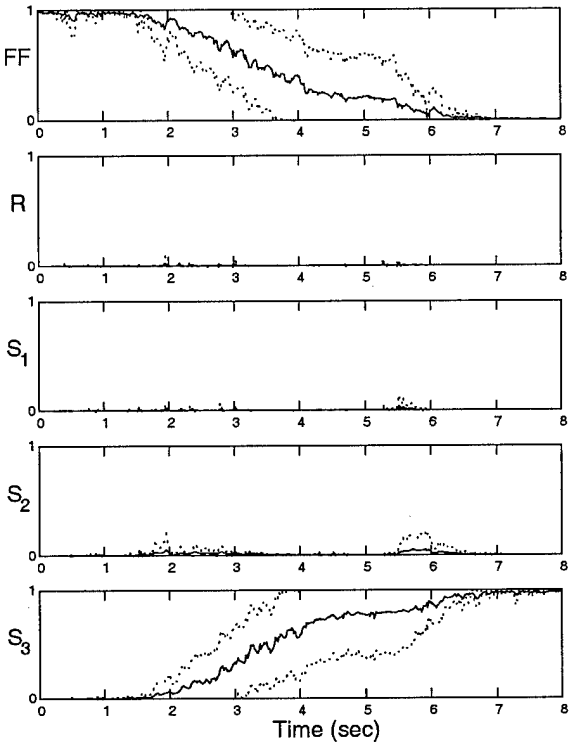


(d) Discretization Set: 40%, 60%, 80%

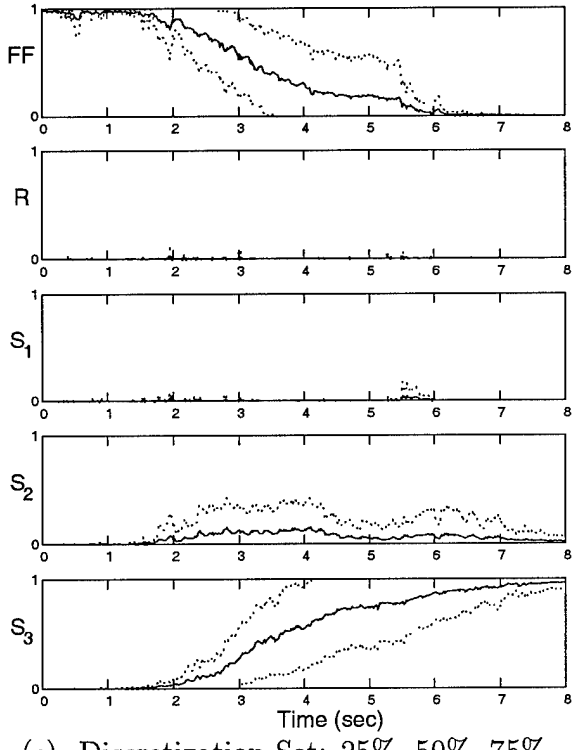
Figure 81. Probability Plot: Rudder Failure, $\epsilon = 80\%$



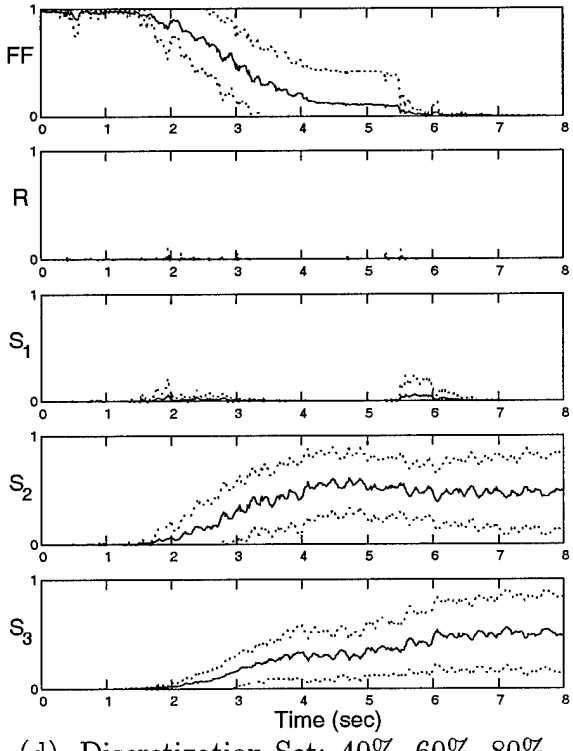
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%

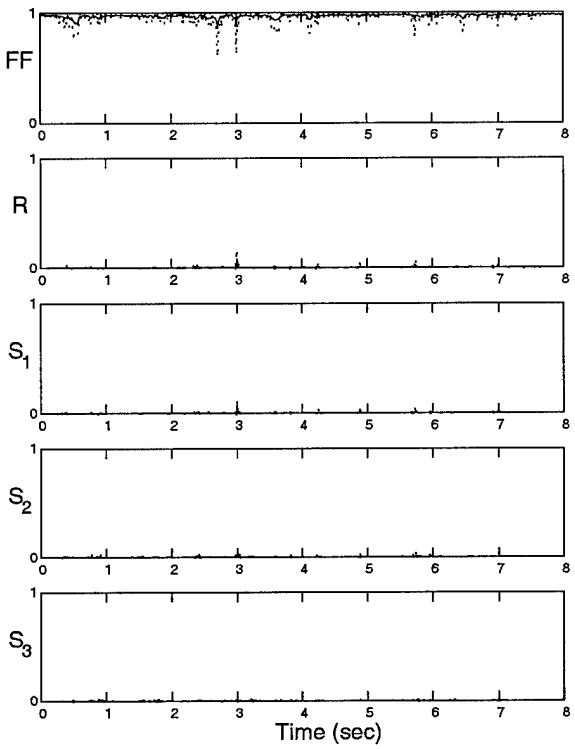


(c) Discretization Set: 25%, 50%, 75%

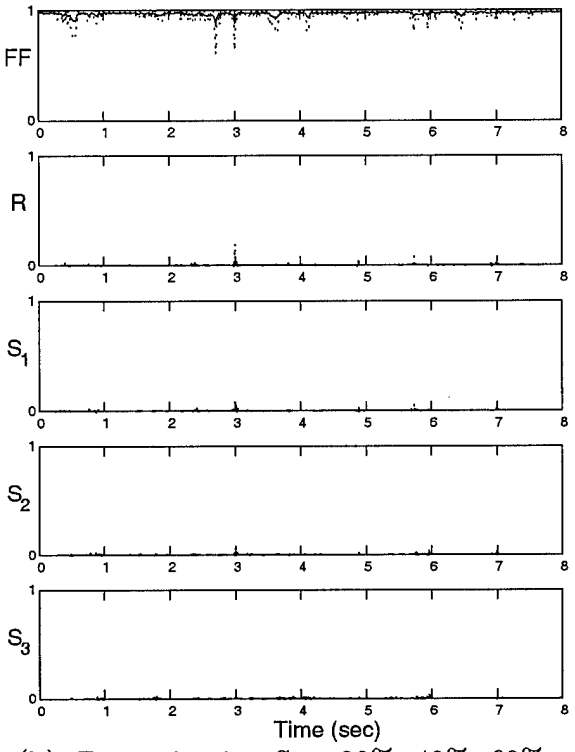


(d) Discretization Set: 40%, 60%, 80%

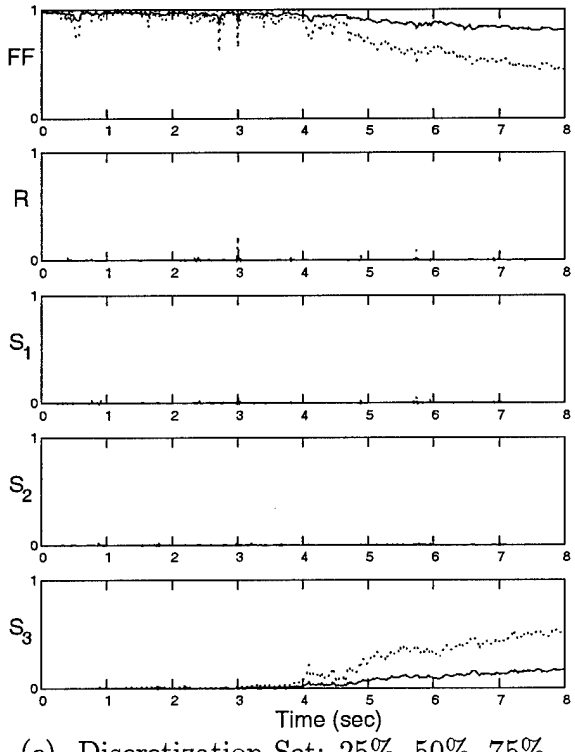
Figure 82. Probability Plot: Rudder Failure, $\epsilon = 90\%$



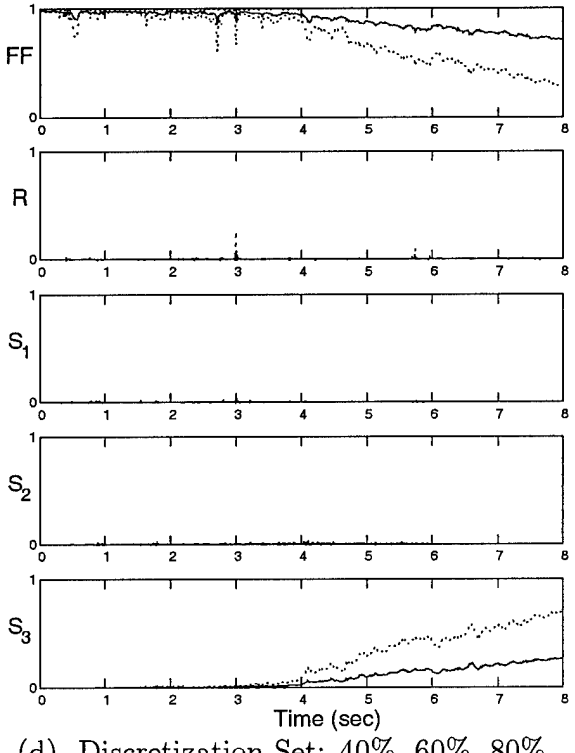
(a) Discretization Set: 10%, 25%, 50%



(b) Discretization Set: 20%, 40%, 60%



(c) Discretization Set: 25%, 50%, 75%



(d) Discretization Set: 40%, 60%, 80%

Figure 83. Probability Plot: Rudder Failure, $\epsilon = 100\%$

APPENDIX C - Parameter Estimate Versus Time Plots

This appendix includes the parameter estimate versus time for the right stabilator, left flaperon, right flaperon, and the rudder. The parameter estimate versus time for the left stabilator was shown in Section 5.4.

C.1 Right Stabilator

The figure numbers and page numbers for the parameter estimate versus time plots for the right stabilator failure are

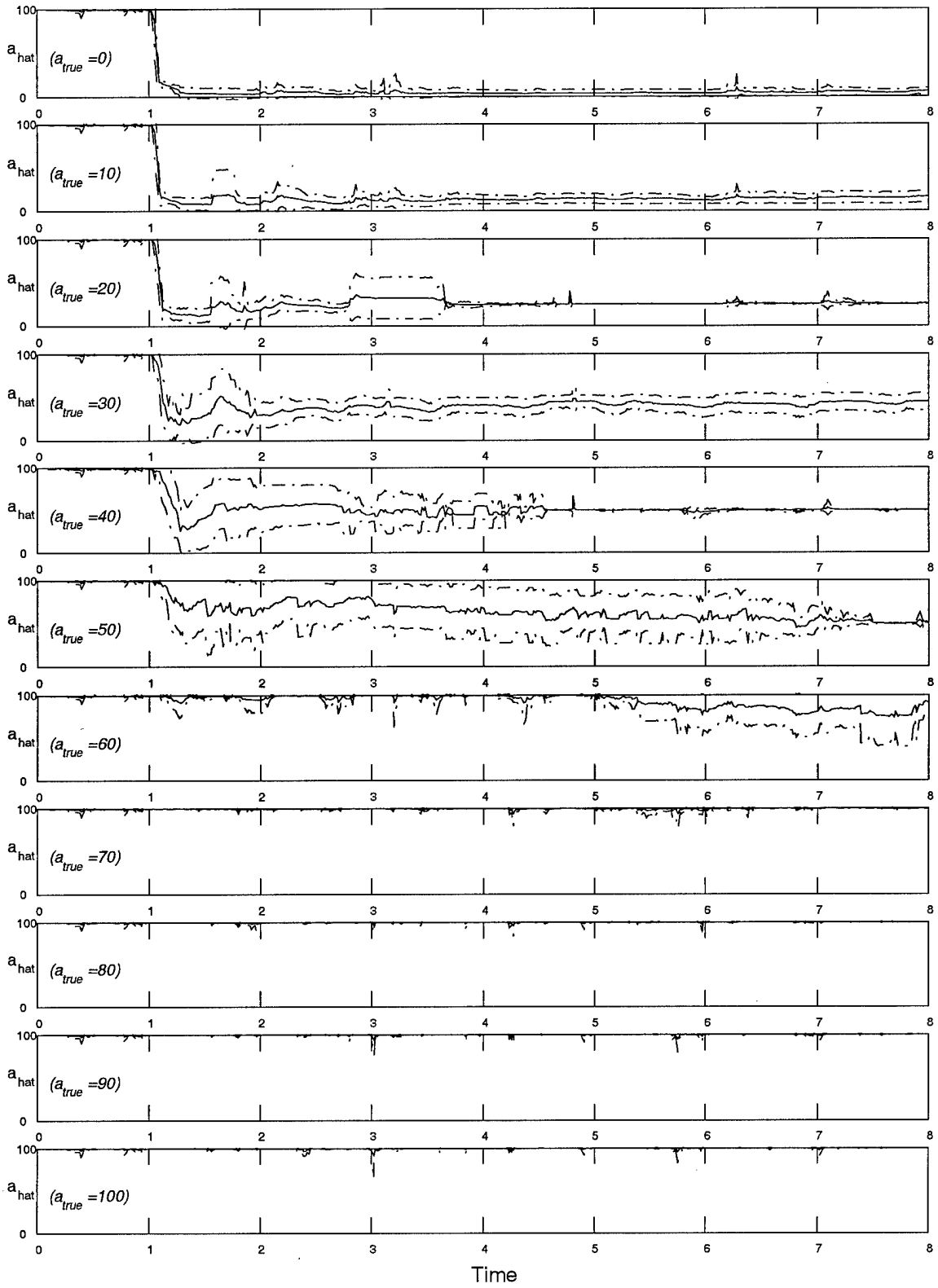


Figure 84. â Versus Time Plots for Right Stabilator Failure Using Spawned Filters 10%, 25%, 50%.

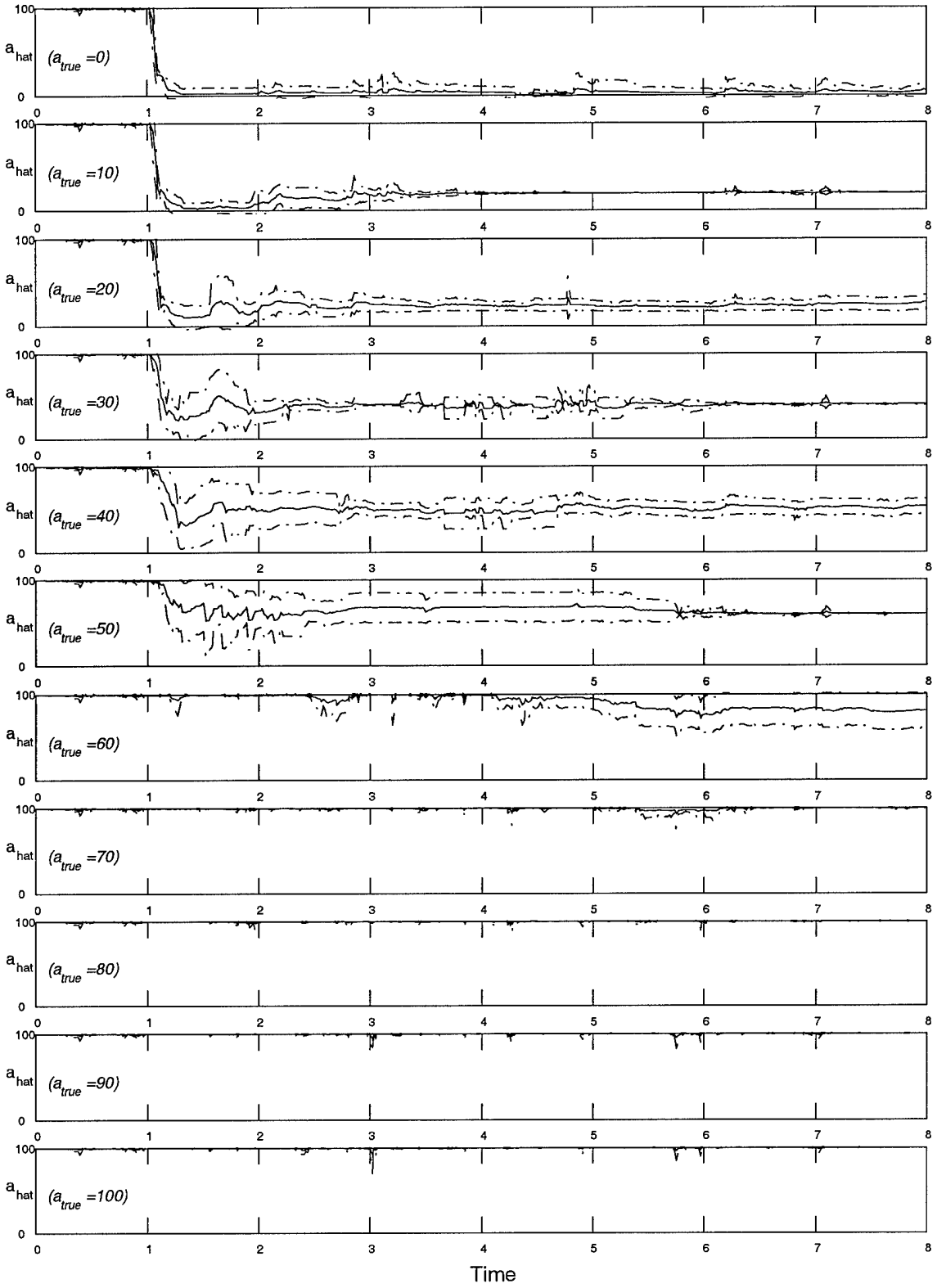


Figure 85. â Versus Time Plots for Right Stabilator Failure Using Spawned Filters 20%, 40%, 60%.

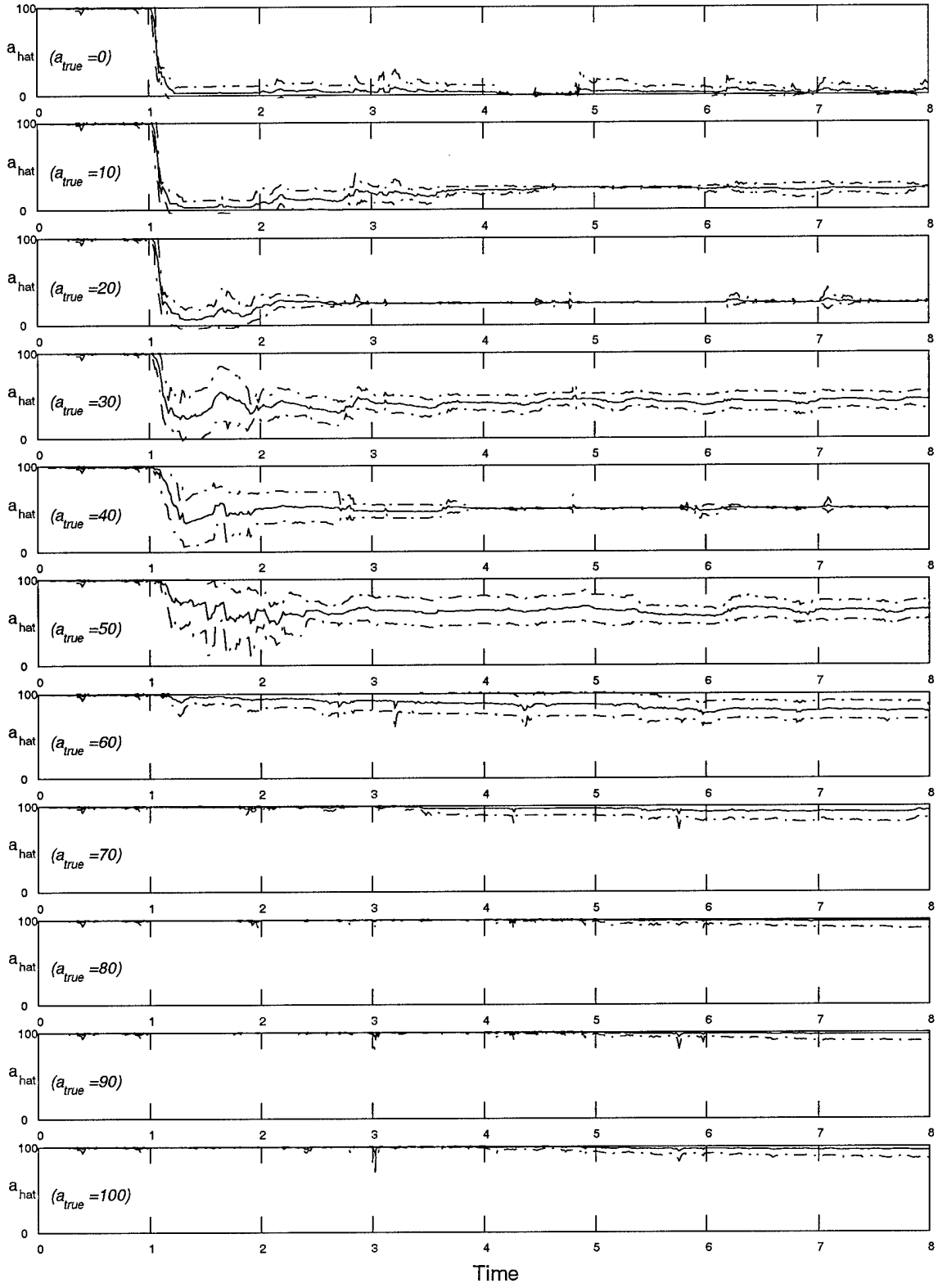


Figure 86. â Versus Time Plots for Right Stabilator Failure Using Spawned Filters 25%, 50%, 75%.

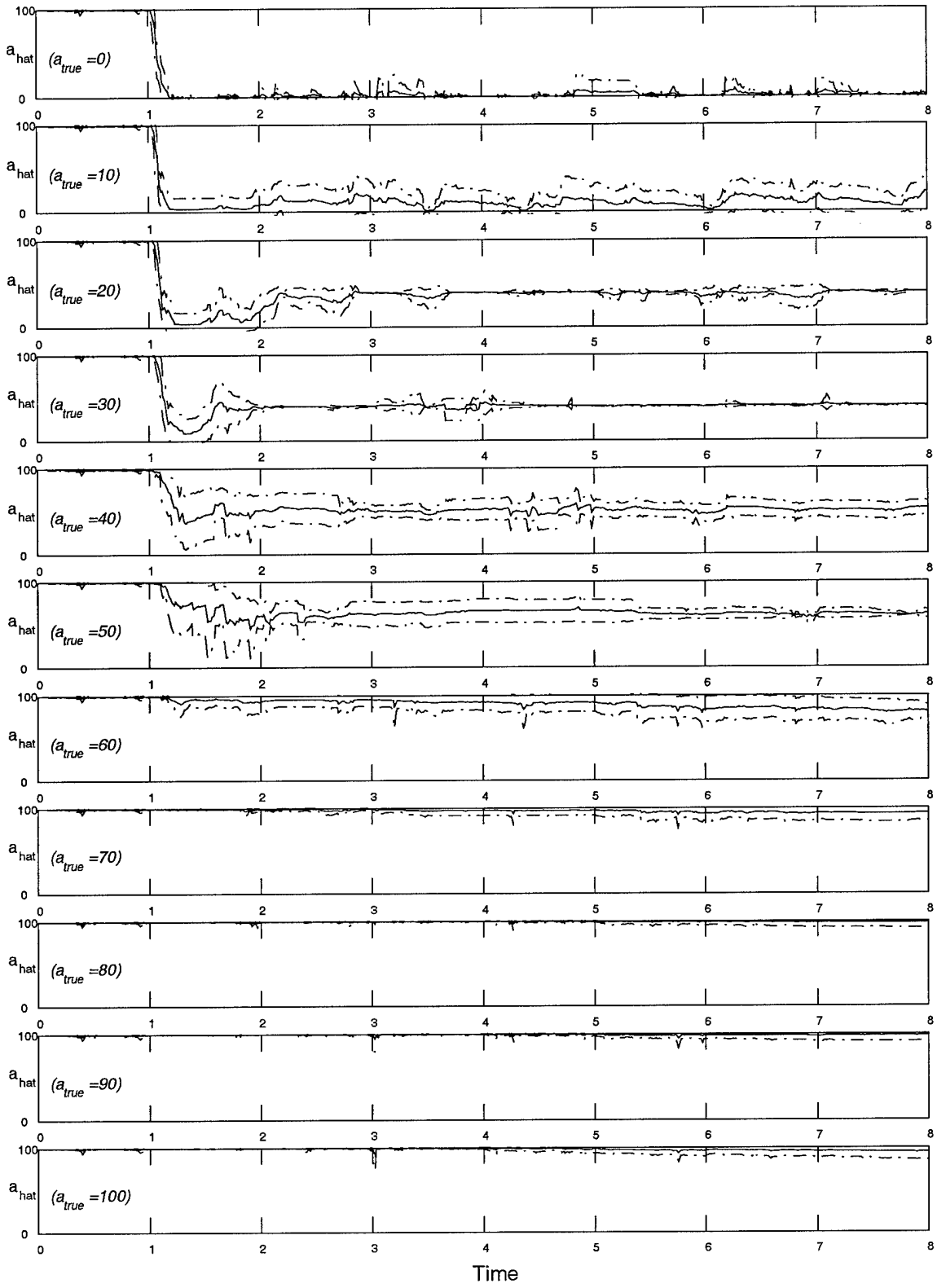


Figure 87. â Versus Time Plots for Right Stabilator Failure Using Spawned Filters 40%, 60%, 80%.

C.2 Left Flaperon

The figure numbers and page numbers for the parameter estimate versus time plots for the right stabilator failure are

Figure	Discretization Set	Page
88	#1 – 10%, 25%, 50%	189
89	#2 – 20%, 40%, 60%	190
90	#3 – 25%, 50%, 75%	191
91	#4 – 40%, 60%, 80%	192

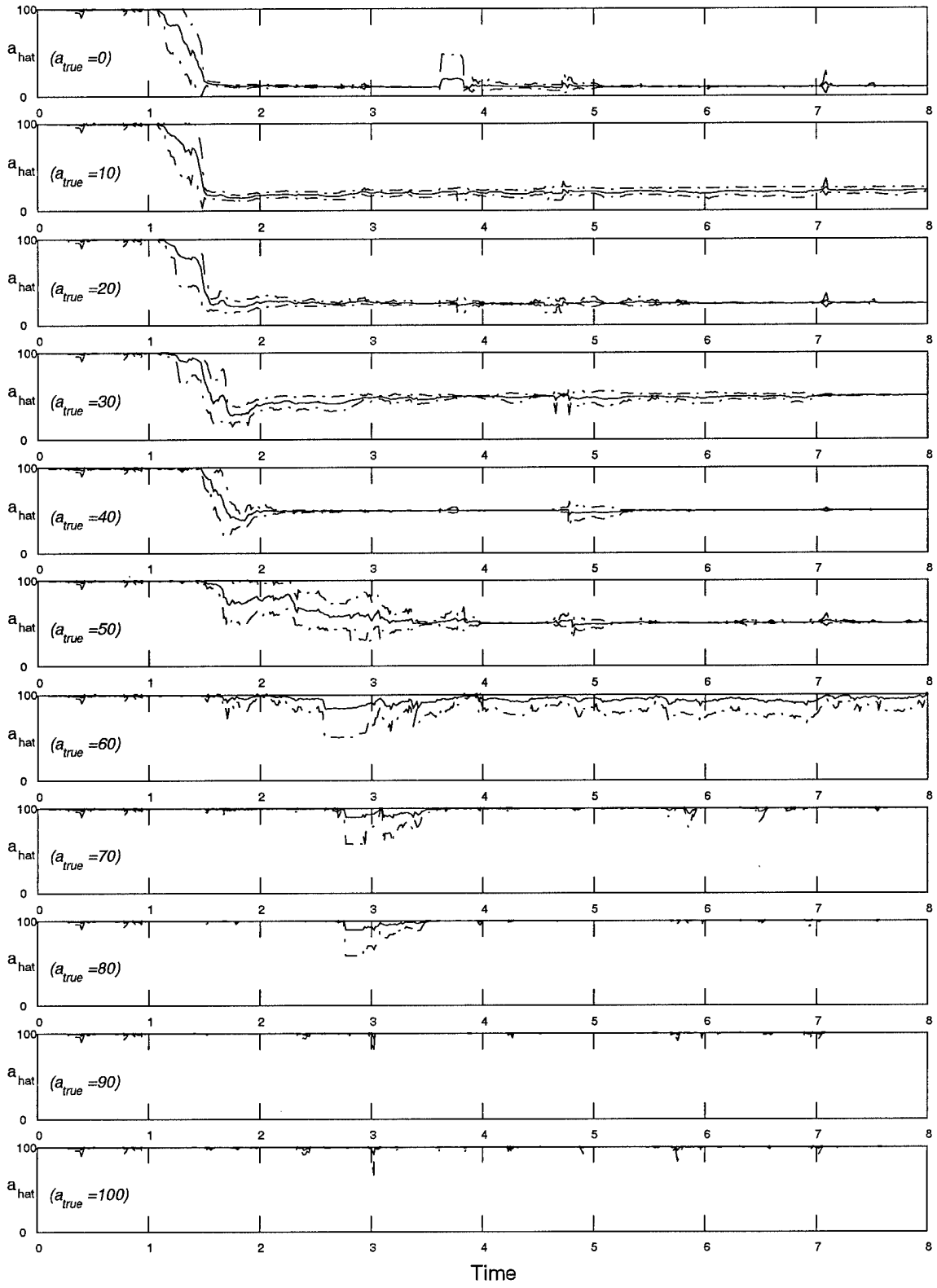


Figure 88. â Versus Time Plots for Left Flaperon Failure Using Spawned Filters 10%, 25%, 50%.

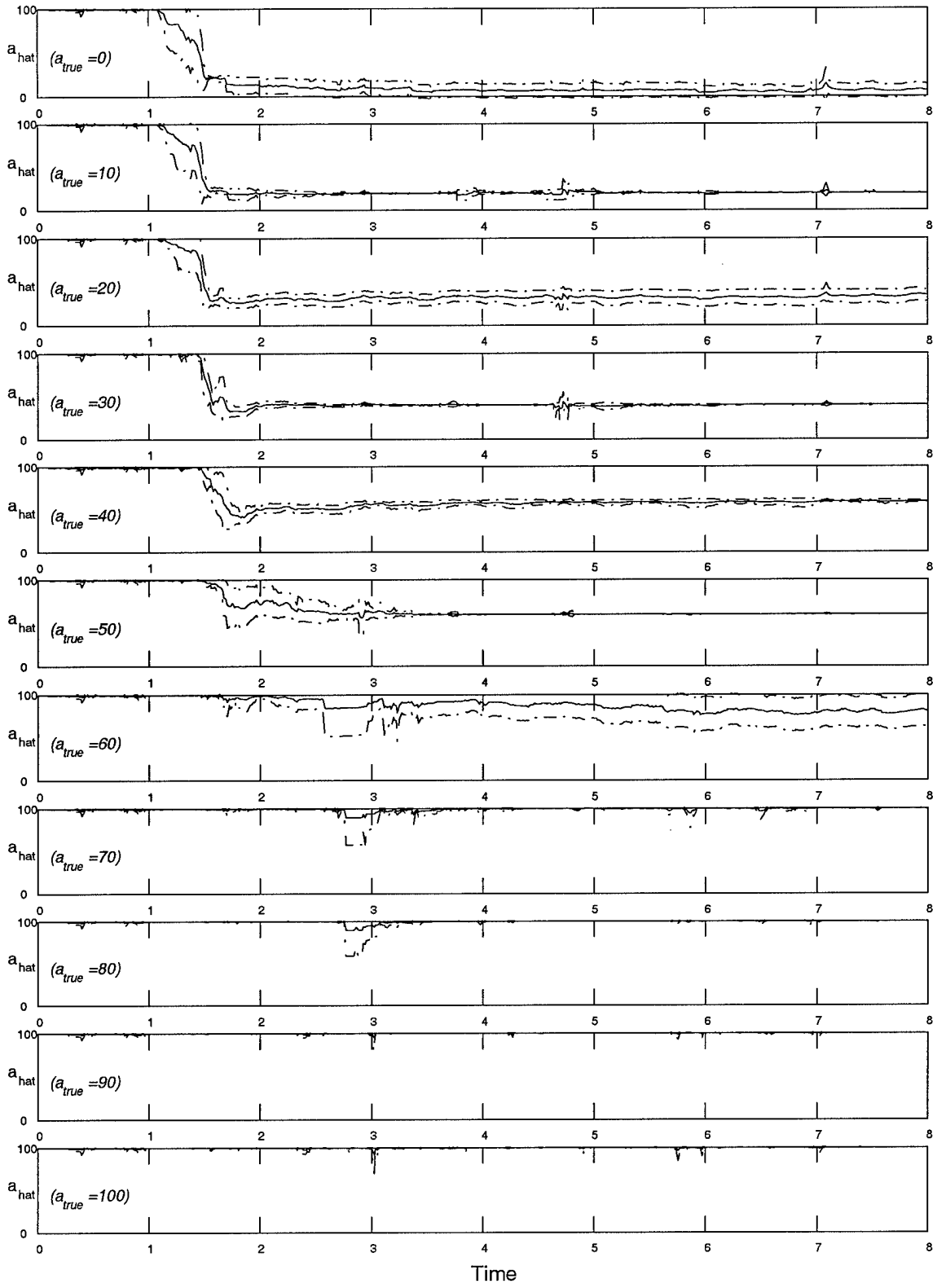


Figure 89. â Versus Time Plots for Left Flaperon Failure Using Spawned Filters 20%, 40%, 60%.

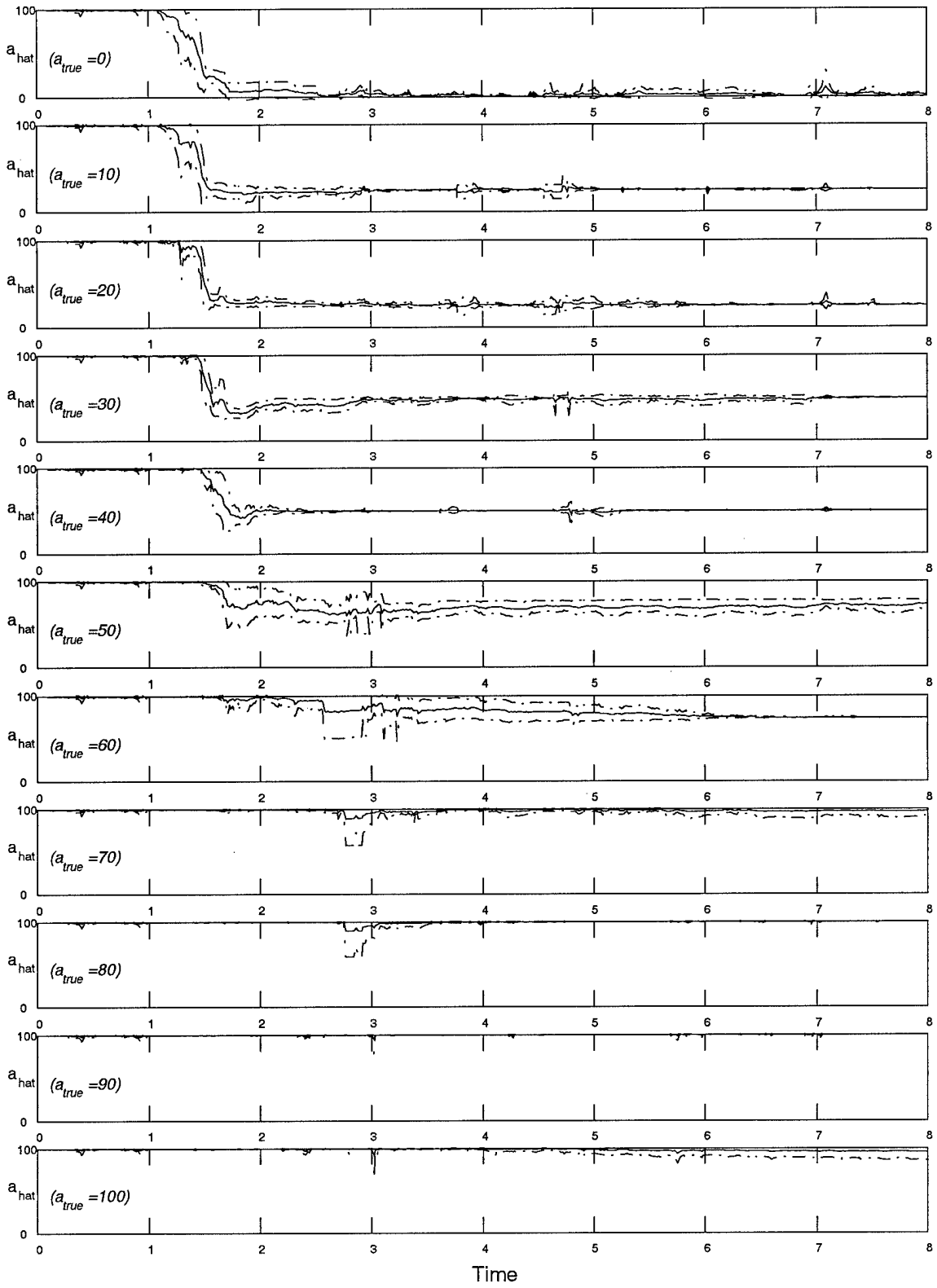


Figure 90. â Versus Time Plots for Left Flaperon Failure Using Spawned Filters 25%, 50%, 75%.

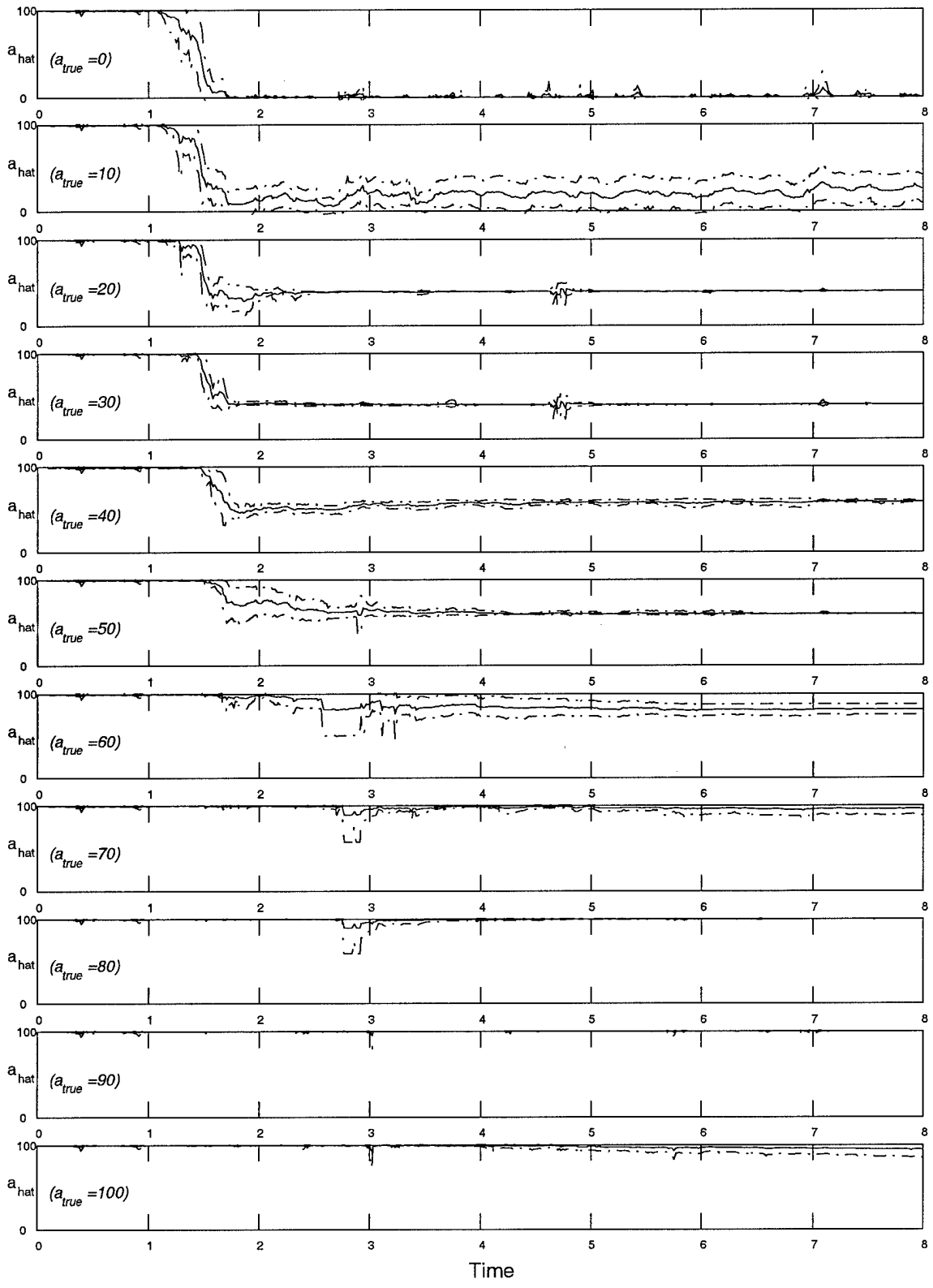


Figure 91. â Versus Time Plots for Left Flaperon Failure Using Spawned Filters 40%, 60%, 80%.

C.3 Right Flaperon

The figure numbers and page numbers for the parameter estimate versus time plots for the right stabilator failure are

Figure	Discretization Set	Page
92	#1 – 10%, 25%, 50%	194
93	#2 – 20%, 40%, 60%	195
94	#3 – 25%, 50%, 75%	196
95	#4 – 40%, 60%, 80%	197

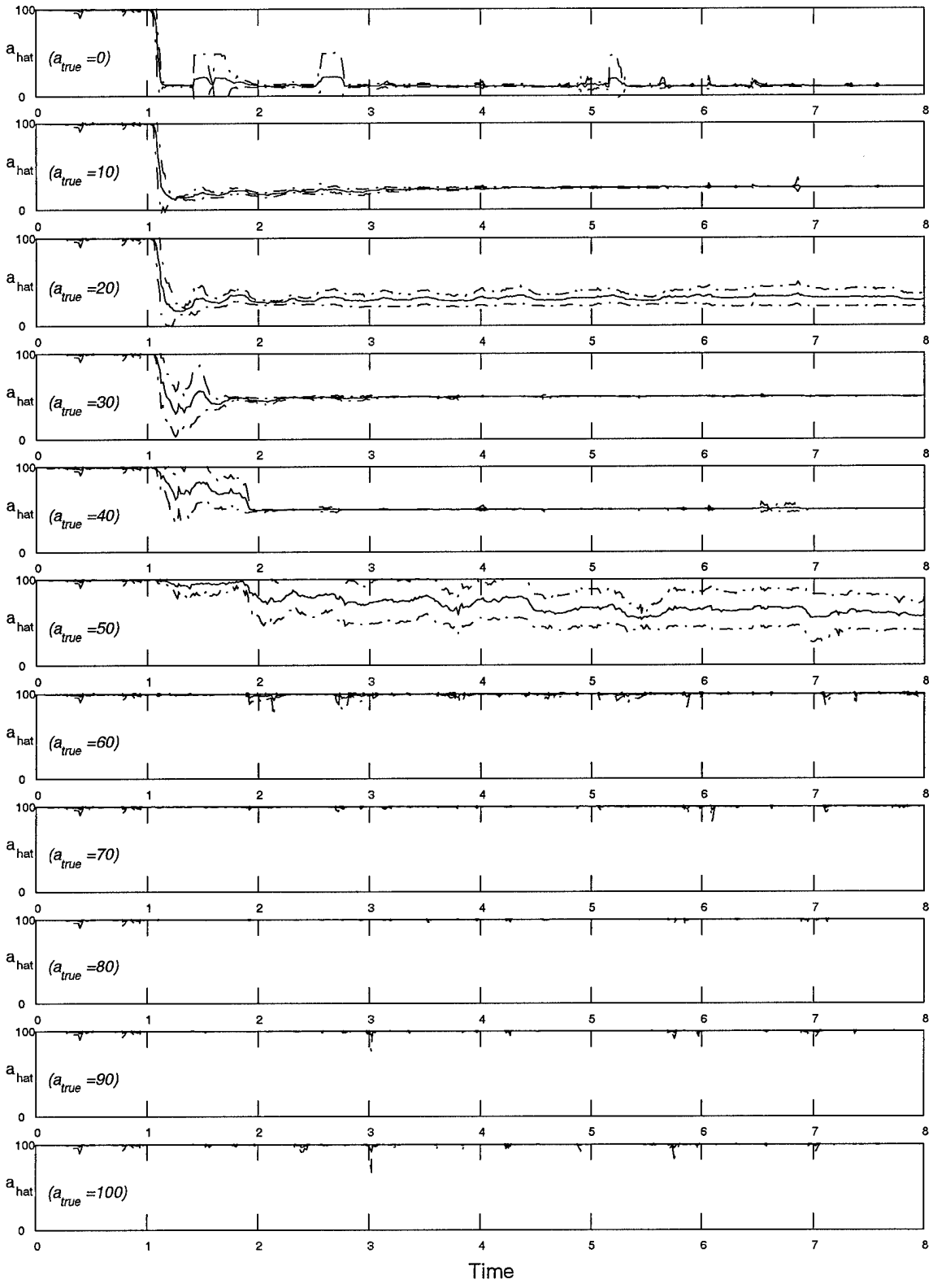


Figure 92. â Versus Time Plots for Right Flaperon Failure Using Spawned Filters 10%, 25%, 50%.

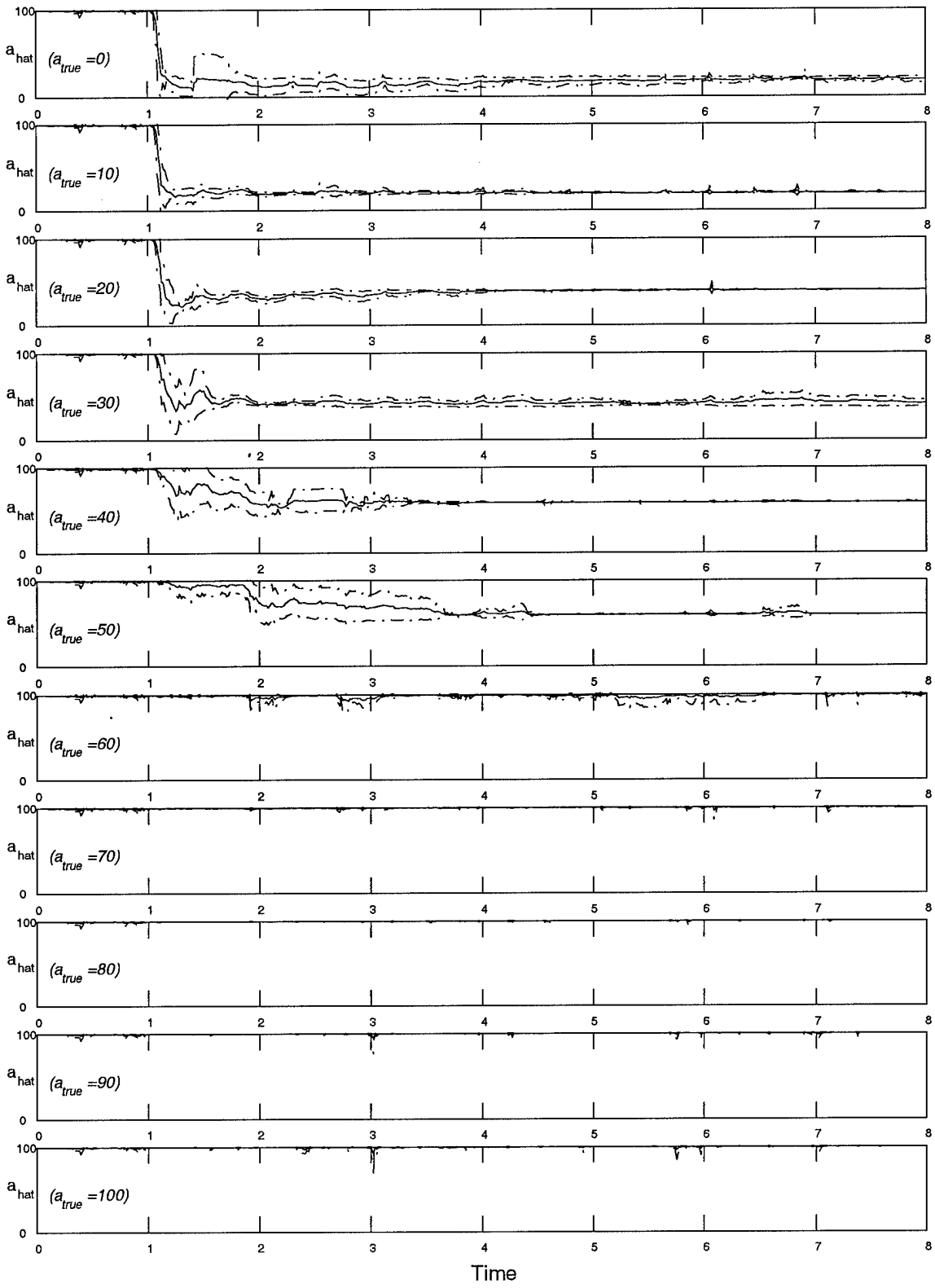


Figure 93. â Versus Time Plots for Right Flaperon Failure Using Spawned Filters 20%, 40%, 60%.

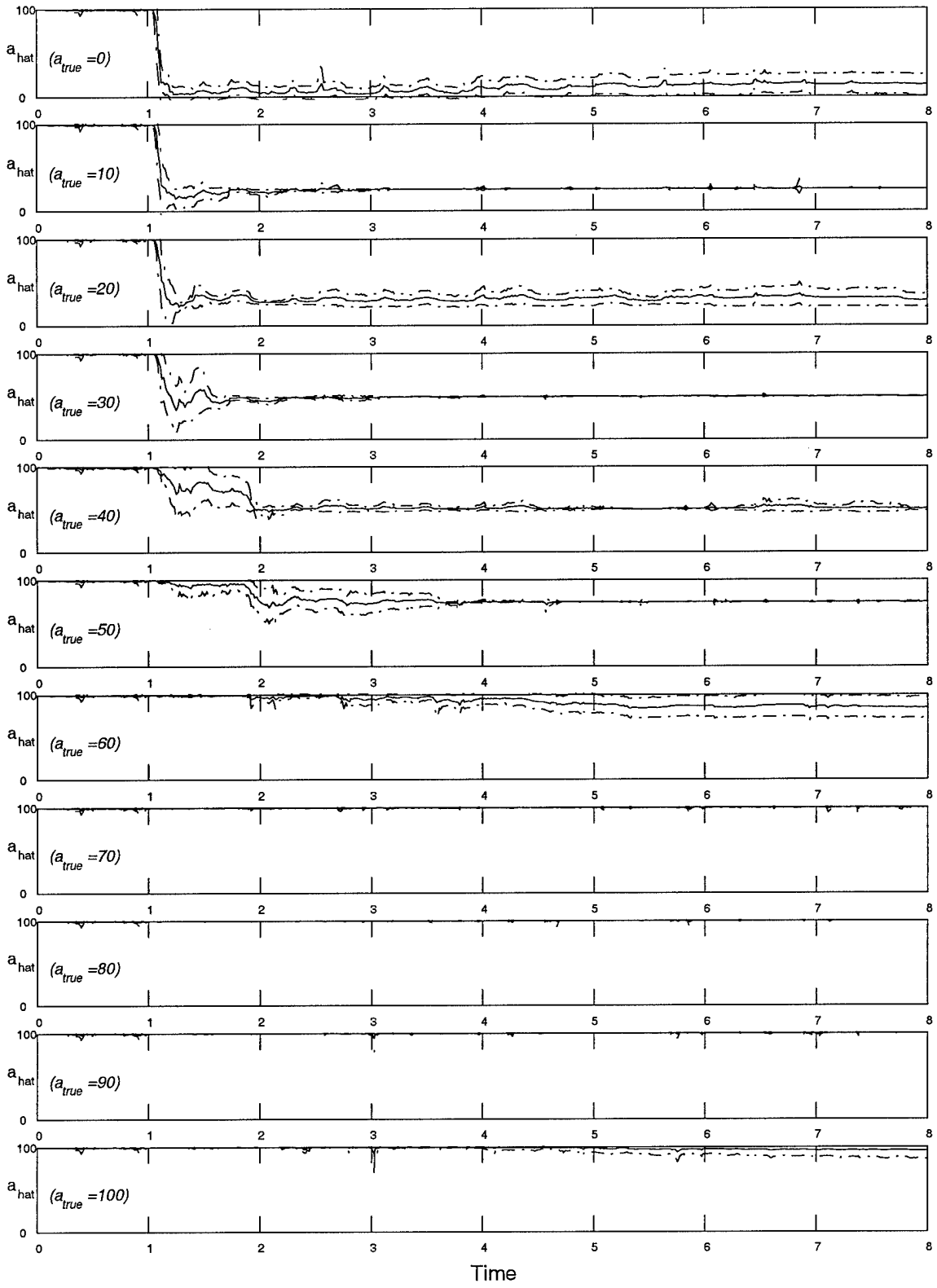


Figure 94. â Versus Time Plots for Right Flaperon Failure Using Spawned Filters 25%, 50%, 75%.

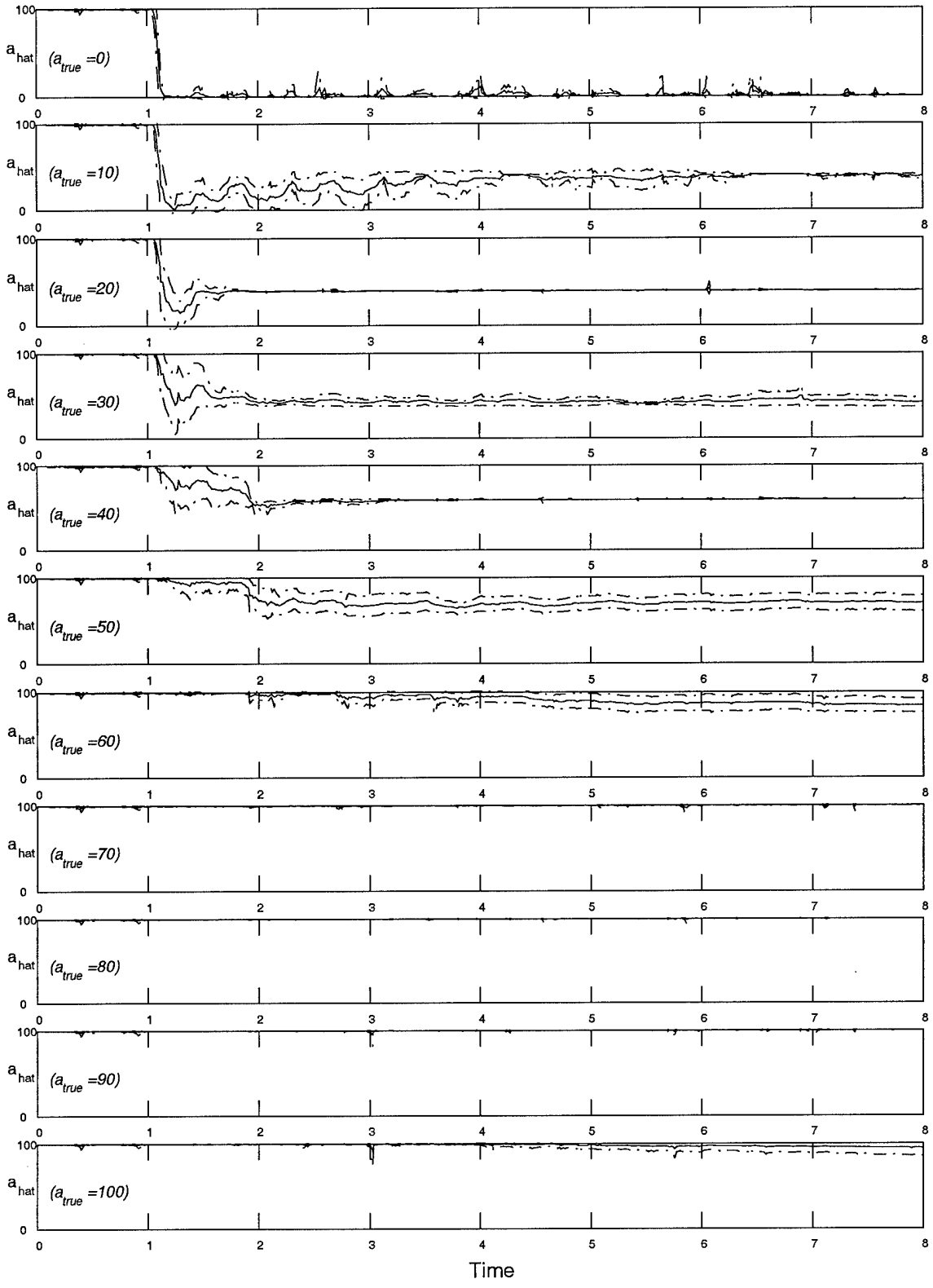


Figure 95. â Versus Time Plots for Right Flaperon Failure Using Spawned Filters 40%, 60%, 80%.

C.4 Rudder

The figure numbers and page numbers for the parameter estimate versus time plots for the right stabilator failure are

Figure	Discretization Set	Page
96	#1 – 10%, 25%, 50%	199
97	#2 – 20%, 40%, 60%	200
98	#3 – 25%, 50%, 75%	201
99	#4 – 40%, 60%, 80%	202

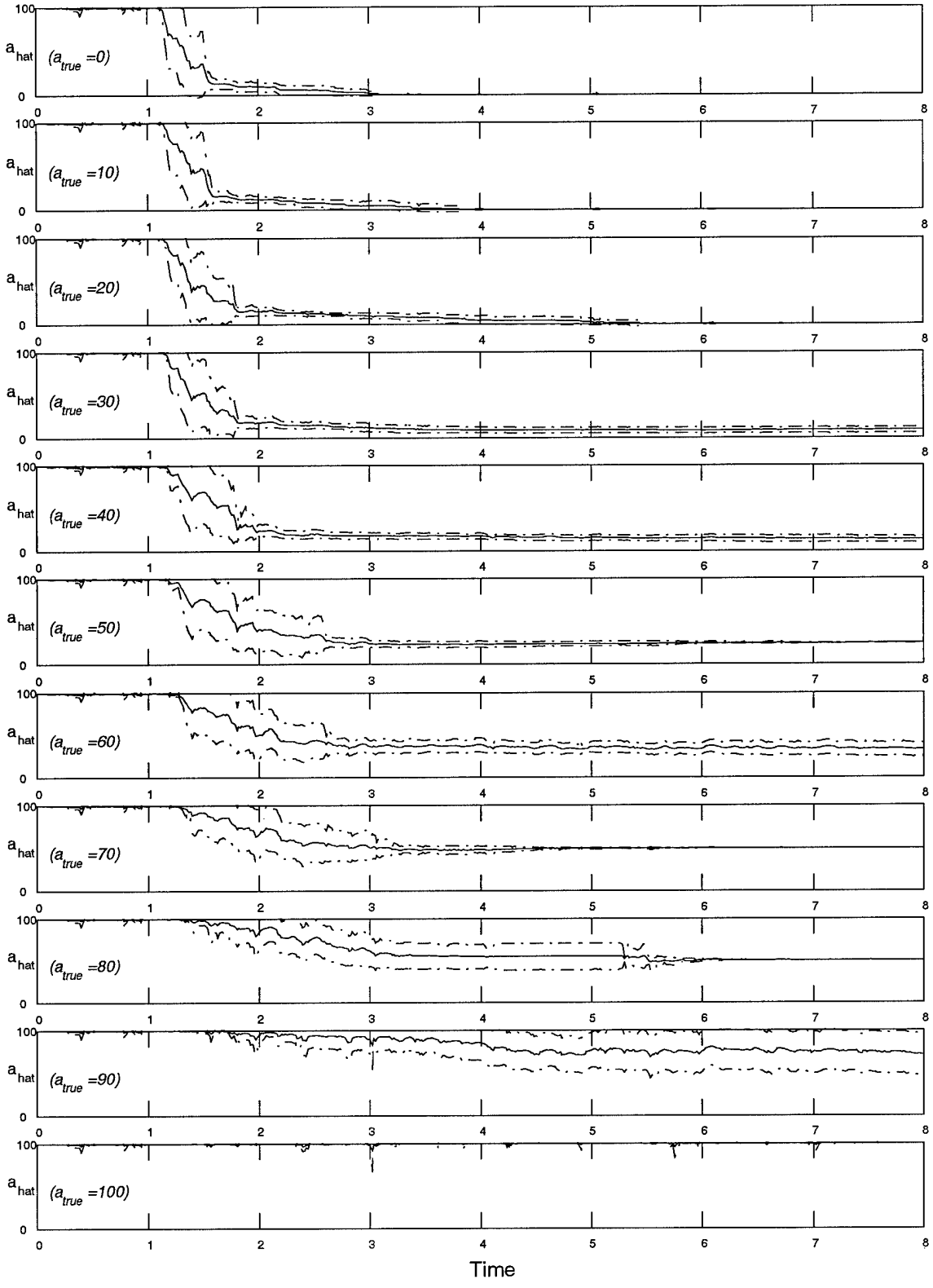


Figure 96. â Versus Time Plots for Rudder Failure Using Spawned Filters 10%, 25%, 50%.

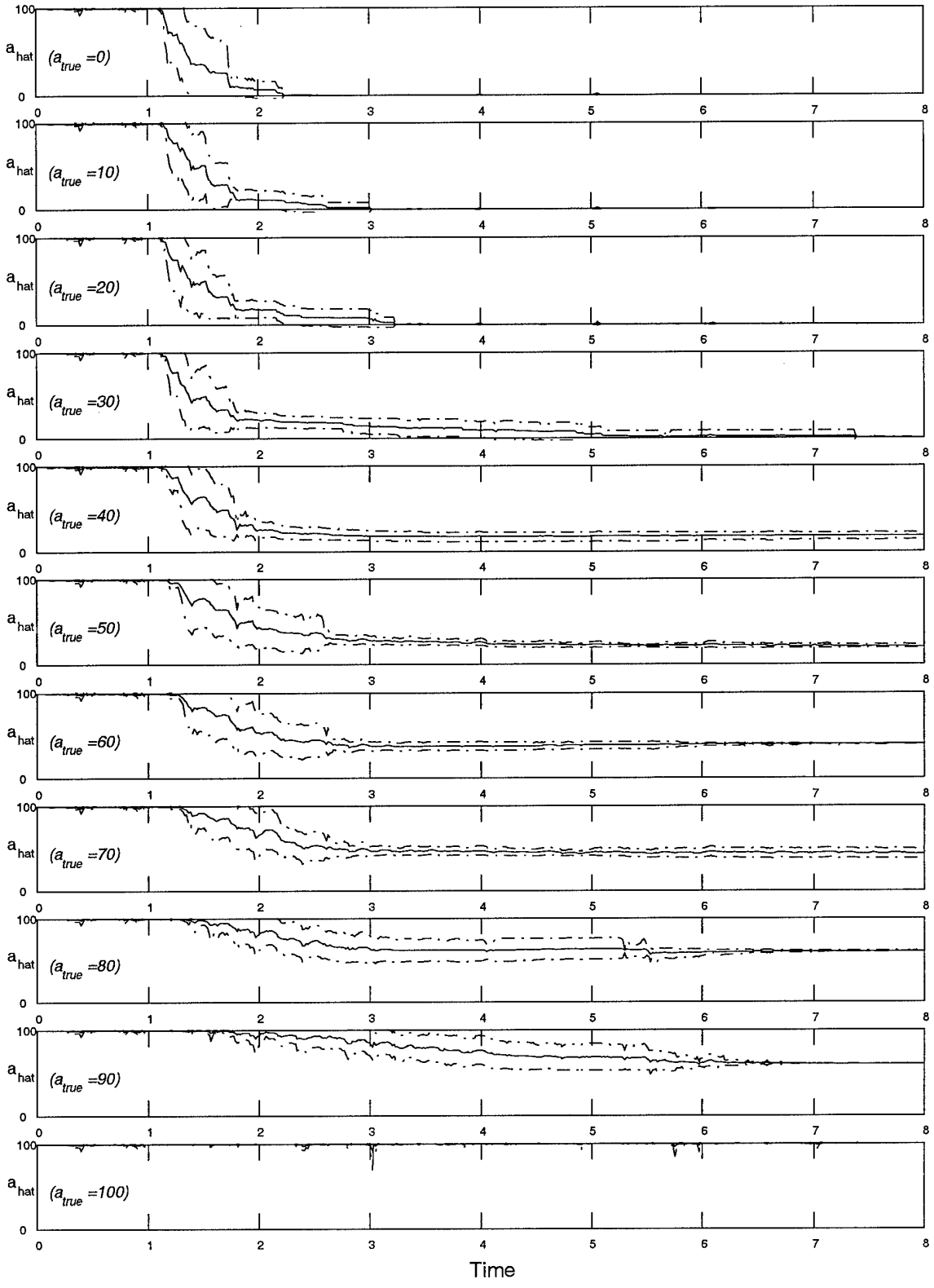


Figure 97. â Versus Time Plots for Rudder Failure Using Spawned Filters 20%, 40%, 60%.

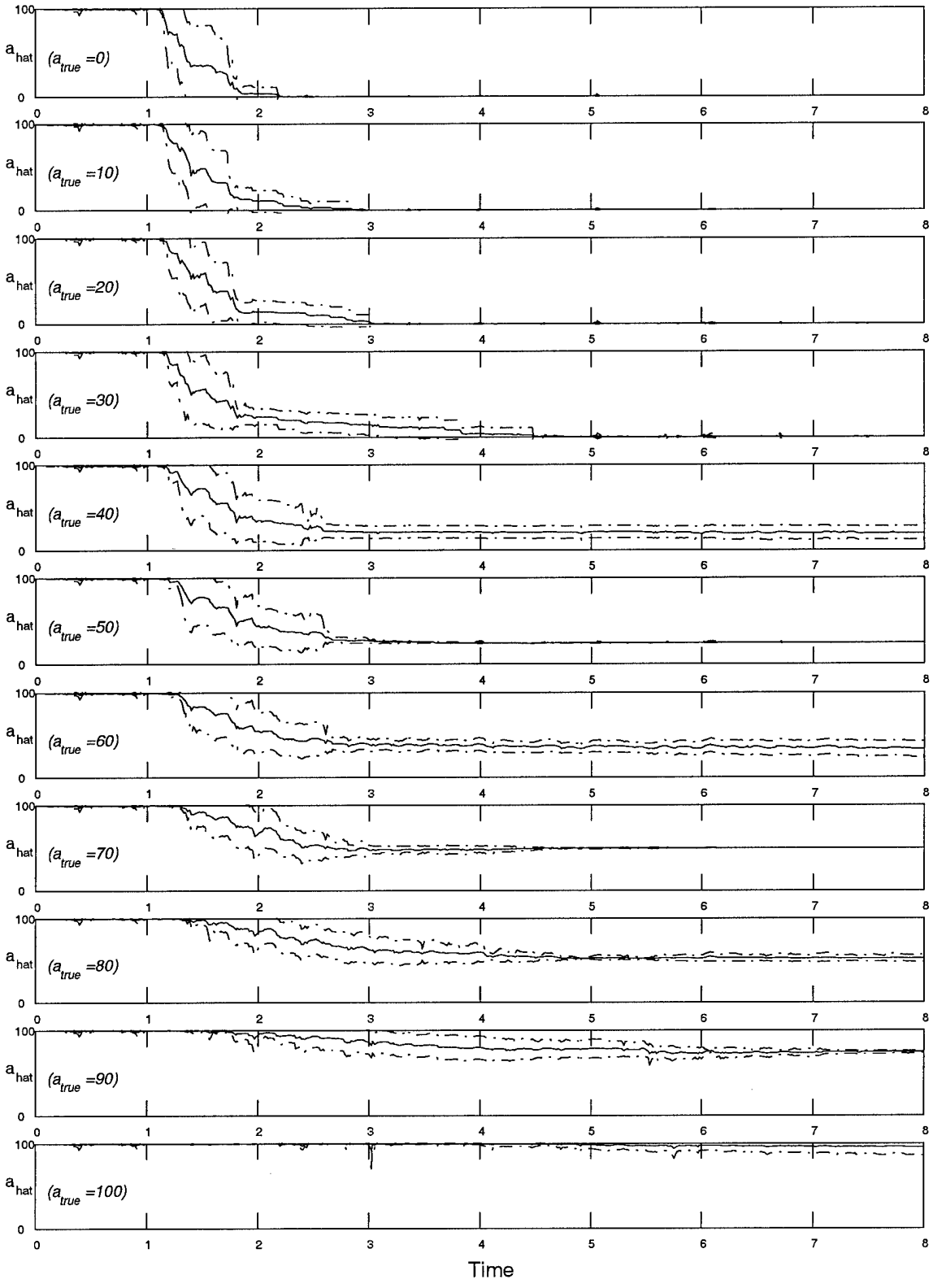


Figure 98. â Versus Time Plots for Rudder Failure Using Spawned Filters 25%, 50%, 75%.

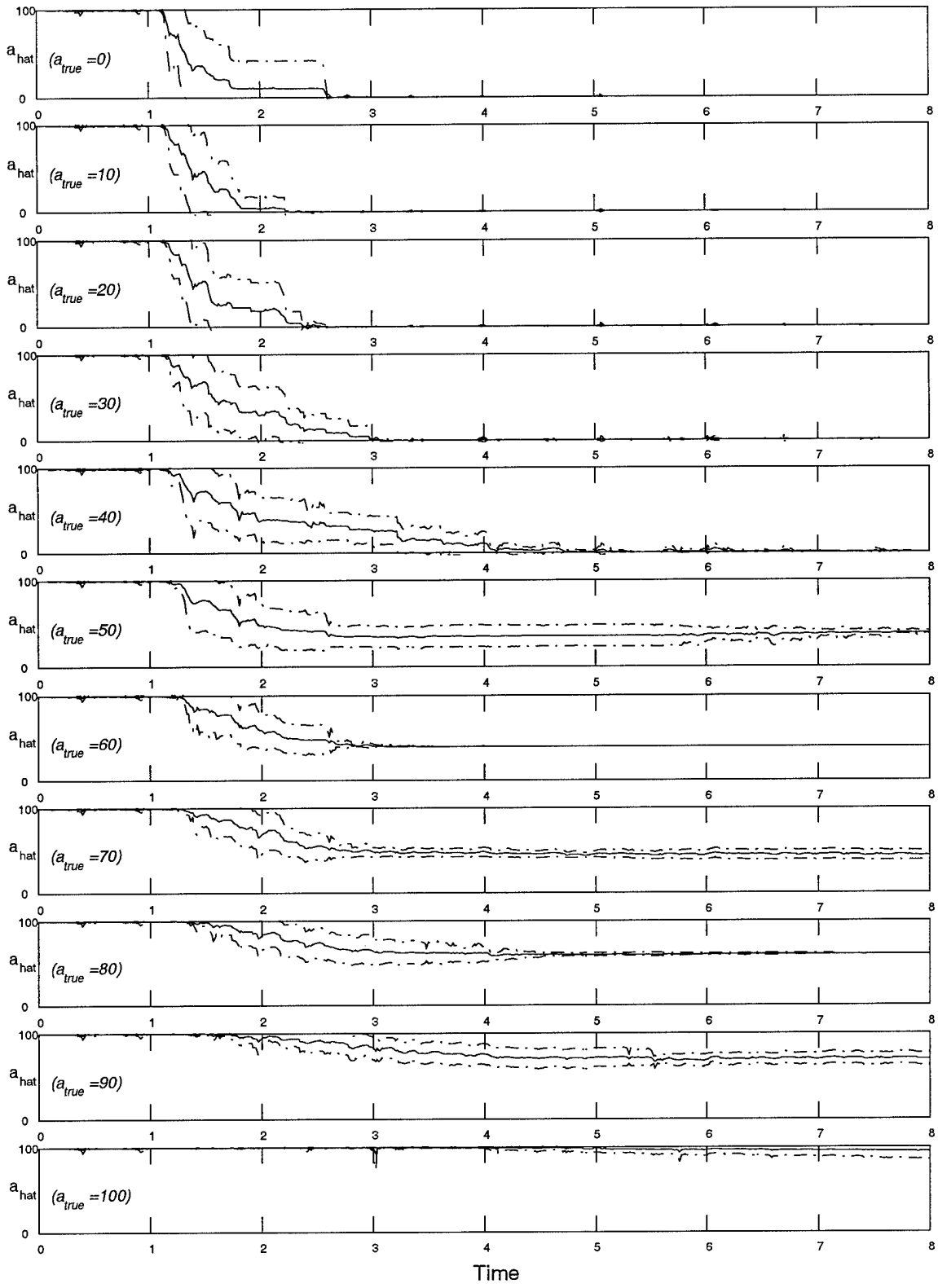


Figure 99. â Versus Time Plots for Rudder Failure Using Spawned Filters 40%, 60%, 80%.

BIBLIOGRAPHY

[1] Athans, M., et al. "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method - Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, AC-22(5):768-780 (October 1977).

[2] Baram, Y., and Sandell, N. R., Jr., An information theoretic approach to dynamic system modelling and identification, *IEEE Transactions on Automatic Control* AC-23(1), 61-66 (1978).

[3] Baram, Y., and Sandell, N. R., Jr., Consistent estimation of finite parameter sets with application to linear systems identification, *IEEE Transactions on Automatic Control* AC-23(3), 451-454 (1978).

[4] Baram, Yoram. *Information, Consistent Estimation and Dynamic System Identification.* PhD Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, November 1976.

[5] Bar-Shalom, Yaakov and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software.* Boston, MA: Artech House, 1993.

[6] Blackelock, John H. Automatic Control of Aircraft and Missles. New York: John Wiley & Sons, Inc., 1991.

[7] Chang, C. B. and M. Athans. "State Estimation for Discrete Systems with Switching Parameters," *IEEE Transactions on Aerospace and Electronic Systems*, AES-14(4):418-424 (May 1978).

[8] Clark, Curtis S. *Multiple Model Adaptive Estimation and Control Redistribution Performance on the VISTA F-16 During Parial Actuator Impairments.* MS Thesis, AFIT/GE/ENG/97D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1997.

[9] Dasgupta, S. and L. C. Westphal. "Convergence of Partitioned Adaptive Filters for Systems with Unknown Biases," *IEEE Transactions on Automatic Control*, 28:614-615 (May 1983).

[10] Eide, Capt Peter Keith. *Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16.* MS Thesis, AFIT/GE/ENG/94D-06, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.

[11] Eide, Peter K. and Peter S. Maybeck. "An MMAE Failure Detection System for the F-16," *IEEE Transactions on Aerospace and Electronic Systems*, 32(3):1125-1136 (July 1996).

[12] Fry, C. M. and A. P. Sage "On Hierarchical Structure Adaption and System Identification," *International Journal of Control*, 20(3):433-452 (September 1974).

[13] Gierke, Henning E. et al. *Stochastic Models, Estimation, and Control, II, Book 1*, 355-405. Washington, DC: Scientific and Technical Information Office, National Aeronautics and Space Administration, 1975.

[14] Gustafson, John A. and Peter S. Maybeck. "Flexible Spacestructure Control Via Moving-Bank Multiple Model Algorithms," *IEEE Transactions on Aerospace and Electronic*

Systems, 30:750-757 (July 1994).

[15] Hawkes, R. M. and J. B. Moore. "Performance Analysis of Bayesian Parameter Estimators," *Proceedings of the IEEE*, 64:1143-1150 (August 1976).

[16] Hentz, K. P. *Feasibility Analysis of Moving Bank Multiple Model Adaptive Estimation and Control Algorithms.* MS Thesis, AFIT/EE/ENG/84D-32, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1984.

[17] Lainiotis, D. G. "Partitioning: A Unifying Framework for Adaptive Systems, I: Estimation," *Proceedings of the IEEE*, 64:1182-1197 (August 1976).

[18] Lewis, Capt Robert W. *Multiple Model Adaptive Estimation and Control Redistribution for the VISTA F-16.* MS Thesis, AFIT/GE/ENG/96D-29, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.

[19] Li, Xiao-Rong and Yaakov Bar-Shalom. "Multiple Model Estimation with Variable Structure," *IEEE Transactions on Automatic Control*, 41(4):479-493 (April 1996).

[20] Li, Xiao-Rong, et. al. "Multiple-Model Estimation with Variable Structure: Model-Group Switching Algorithm," *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, CA, 3114-3119 (December 1997).

[21] Magill, D. T. "Optimal Adaptive Estimation of Sample Stochastic Processes," *IEEE Conference on Decision and Control*, AC-10(4):434-439 (October 1965).

[22] Maybeck, Peter S. *Stochastic Models, Estimation, and Control, I.* New York: Academic Press, Inc., 1979. Republished - Arlington, VA: Navtech, 1994.

[23] Maybeck, Peter S. *Stochastic Models, Estimation, and Control, II.* New York: Academic Press, Inc., 1982. Republished - Arlington, VA: Navtech, 1994.

[24] Maybeck, Peter S. and Donald L. Pogoda. "Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures," In *Proceedings of the 28th Conference on Decision and Control*, 1566-1572 (December 1989).

[25] Maybeck, Peter S. and Karl P. Hentz. "Investigation of Moving-Bank Multiple Model Adaptive Algorithms," *Journal of Guidance and Control*, 10(1):90-96 (January-February 1987).

[26] Maybeck, Peter S. and Michael R. Schore. "Reduced-Order Multiple Model Adaptive Controller for Flexible Spacestructure," *IEEE Transactions on Aerospace and Electronic Systems*, 28:756-767 (July 1992).

[27] Maybeck, Peter S. and P. D. Hanlon. "Performance Enhancement of a Multiple Model Adaptive Estimator," *IEEE Transactions on Aerospace and Electronic Systems*, 31(4):1240-1254 (October 1995).

[28] Maybeck, Peter S. and R. D. Stevens. "Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods," *IEEE Transactions on Aerospace and Electronic Systems*, AES-27(3):470-480 (May 1991).

[29] Maybeck, P. S. and R. I. Suizu. "Adaptive Tracker Field of View Variation Via Multiple

Model Filtering," *IEEE Transactions on Aerospace and Electronic Systems*, 21(4):529-537 (July 1985).

[30] Maybeck, Peter S., et al. "Target Tracking Using Infrared Measurements and Laser Illumination," *IEEE Transactions on Aerospace and Electronic Systems*, 30:758-768 (July 1994).

[31] Menke, Timothy E. *Multiple Model Adaptive Estimation Applied to the VISTA F-16 with Actuator and Sensor Failiures.* MS Thesis, AFIT/GA/ENG/92J-01, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 1992.

[32] Menke, Timothy E. and Peter S. Maybeck. "Sensor/Actuator Failure Detection in the VISTA F-16 by Multiple Model Adaptive Estimation," *IEEE Transactions on Aerospace and Electronic Systems,* 31(4):1218-1229 (October 1995).

[33] *MIL-STD-1797A Flying Qualities of Piloted Aircraft*, January 1990.

[34] Moore, J. B. and R. M. Hawkes. "Decision Methods in Dynamic System Identification." *Proceedings of the IEEE Conference on Decision and Control*, 14:645-650 (1975).

[35] Nesline, F. W., Wells, B. H., and Zarchan, P., "Combined Optimal/Classical Approach to Robust Missle Autopilot Design," *Journal of Guidance and Control*, Vol. 4:316-322 (May-June 1981).

[36] Phillips, Maj. Scott N. *A Quantitative Feedback Theory FCS Design for the Subsonic Envelope of the VISTA F-16 Including Configuration Variation.* MS Thesis, AFIT/GE/ENG/88D-24, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.

[37] Pogoda, Capt Donald L. *Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures.* MS Thesis, AFIT/GE/ENG/88D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1988.

[38] Schaefer, Karl E. *Bioastronautics.* New York: The Macmillan Company, 1964.

[39] Sheldon, Stuart N. *An Optimal Parameter Discretization Strategy for Multiple Model Adaptive Estimation and Control.* PhD Dissertation, AFIT/DS/ENG/89-2, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.

[40] Sheldon, Stuart N. and Peter S. Maybeck. "An Optimizing Design Strategy for Multiple Model Adaptive Estimation and Control," *IEEE Transactions on Automatic Control*, 38(4):651-654 (April 1993).

[41] Stepaniak, 2Lt M. J. *Multiple Model Adaptive Control of the VISTA F-16.* MS Thesis, AFIT/GE/ENG/95D-04, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.

[42] Stepaniak, Michael J. and Peter S. Maybeck. "MMAE-Based Control Redistribution Applied to the VISTA F-16," *IEEE Transactions on Aerospace and Electronic Systems*, 30(4):1249-1260 (October 1998).

[43] Stevens, Capt Richard D. *Characterization of a Reconfigurable Multiple Model Adaptive Controller Using A STOL F-15 Model.* MS Thesis, AFIT/GE/ENG/89D-52, School of

Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.

[44] Strang, G., *Linear Algebra and Its Applications (Third Edition).* San Diego: Harcourt Brace Jovanovich, Publishers, 1988.

[45] Tellman, Larry. *Multiple Model-Based Robot Control: Development and Initial Evaluation.* MS Thesis, AFIT/GE/ENG/88D-55, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1988.

[46] Tugnait, J. K. "Comments on "State Estimation for Escrete Systems with Switching Parameters"," *IEEE Transactions on Aerospace and Electronic Systems*, AES-15(3):464 (May 1979).

[47] Vasquez, Capt Juan R. *New Algorithms for Moving-Bank Multiple Model Adaptive Estimation.* PhD Dissertation, AFIT/GE/ENG/98D-??, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1998.

[48] White, 2Lt Nathan A. *MMAE Detection of Interference/Jamming and Spoofing in a DGPS-Aided Inertial System.* MS Thesis, AFIT/GE/ENG/96D-21, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.

Vita

Second Lieutenant Kenneth A. Fisher was born in Pittsbugh, PA on August 3, 1975. In 1993, he entered Ohio Northern University where he received the degree of Bachelor of Science in Electrical Engineering, graduating in May 1997. He received his commission through the Air Force ROTC program in May 1997 and reported directly to the Air Force Institute of Technology (AFIT). He is pursuing a Masters of Science in Electrical Engineering at the AFIT. Upon completion in March 1999, he will work for the National Air Intelligence Center (NAIC), Wright-Patterson AFB, OH.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Master's Thesis

4. TITLE AND SUBTITLE
Multiple Model Adaptive Estimation Using Filter Spawning

5. FUNDING NUMBERS

6. AUTHOR(S)
Kenneth A. Fisher
Second Lieutenant, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2950 P Street
Wright-Patterson AFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/99M-09

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Captain Odell Reynolds
AFRL/VACC, BLDG 145
Wright-Patterson AFB, OH 45433-7521

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
POC: Dr. Peter S. Maybeck, (937) 255-3636x4639, Peter.Maybeck@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Multiple Model Adaptive Estimation with Filter Spawning is used to detect and estimate partial actuator failures on the VISTA F-16. The truth model is a full six-degree-of-freedom simulation provided by Calspan and General Dynamics. The design models are chosen as 13-state linearized models, including first order actuator models. Actuator failures are incorporated into the truth model and design model assuming a "failure to free stream". Filter Spawning is used to include additional filters with partial actuator failure hypotheses into the Multiple Model Adaptive Estimation (MMAE) bank. The spawned filters are based on varying degrees of partial failures (in terms of effectiveness) associated with the complete-actuator-failure hypothesis with the highest conditional probability of correctness at the current time. Thus, a blended estimate of the failure effectiveness is found using the filters' estimates based upon a no-failure hypothesis (or, an effectiveness of 100%), a complete actuator failure hypothesis (or, an effectiveness of 0%), and the spawned filters' partial-failure hypotheses. This yields substantial precision in effectiveness estimation, compared to what is possible without spawning additional filters, making partial failure adaptation a viable methodology in a manner heretofore unachieved.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Multiple Model Adaptive Estimation, Kalman Filter, Adaptive Flight Control, Actuator Failure Detection	228
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL